1997

A scheduling heuristic for dual bridge cranes in a job shop

Jaime M. Bustos
Lehigh University

Bustos, Jaime M.

A Scheduling Heuristic for Dual Bridge Cranes in a Job Shop

January 12, 1997

A Scheduling Heuristic for Dual Bridge Cranes in a Job Shop

by

Jaime M. Bustos

A Thesis

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Industrial Engineering

Lehigh University

December 1996

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

12-3-96

Date

Thesis Advisor

Thesis Advisor

Chairperson of Department

To my beloved wife Martha

and our son Marcelo

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to my advisors, Dr. S. David Wu and Dr. Robert Storer, for their permanent contributions of ideas and suggestions during the development of this thesis.

Special thanks to the foremen in Machine Shop #1 at BethForge, whose experience helped me in understanding the many issues in the crane scheduling problem.

A sincere gratitude to my colleagues of the Manufacturing Logistics Institute, under which this research was developed, in special to my friend Erhan Kutanoglu for his interesting comments and continuos support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

We study the scheduling of bridge cranes in a jobshop environment. The problem is common in heavy industry settings such as manufacturing of steel rolls where parts are moved around the shop using overhead cranes. A heuristic search approach is proposed which first generates feasible pick-up sequences and then uses a branch and bound procedure to resolve crane collisions resulting from each sequence. A Collision detection method is developed which uses a graphical approach based on time-way diagrams. Minimization of completion time for all the crane moves is considered for deterministic single and multi-stage problems. Random problems have been created to study the performance of the scheduling heuristic. A single-pass procedure is used as a benchmark. For small problems optimal solutions are generated as benchmark. Computational results show that the heuristic provides high quality results when compared with benchmark in all the cases studied. Further improvements are suggested to overcome the explosion of computational effort associated with the branch and bound procedure.

Chapter 1 - INTRODUCTION

Material handling is one of the essential components in a manufacturing. Scheduling material handling devices is important when the material handler is the main bottleneck resource in the system. Special considerations arise when the devices, as in the case of bridge cranes, share a common track or rail. Usually, the scheduling of bridge cranes appears as a problem embedded in shop scheduling. In this case, a global schedule proposed for the shop, normally found without explicitly considering the material handling devices, imposes constraints to the crane scheduling problem (CSP). A typical assumption in these cases is to include the material handling times as part of the actual machining times, or in some cases, completely ignored. In practice, it is common to leave the crane scheduling problem to floor planners who make ad-hoc decisions.

1.1 The Machine Shop #1 At BethForge Inc.

This research is inspired by the crane scheduling problem that arises at Machine Shop #1 at BethForge Inc., a manufacturer of large steel forgings. Figure 1.1 depicts the approximate layout of the machine shop, consisting of two parallel bays each with two overhead cranes. Cranes are used for moving parts between machines and storage areas, download/load parts from/to trucks, setup tools and fixtures in the machines, empty chip boxes generated as a result of the machining operations, and other miscellaneous activities such as moving tools and auxiliary equipment around the shop. The hoist attached to the bridge can move laterally to access machines across the

width of the bay. Because of the common track in use, cranes can never "pass" each other.

Because of the routing characteristics of each product being processed in the shop, the configuration of #1 Machine Shop is essentially a general jobshop, with input and output at one end of the bays. Parts move from one bay to the other using two special purpose transfer carts.

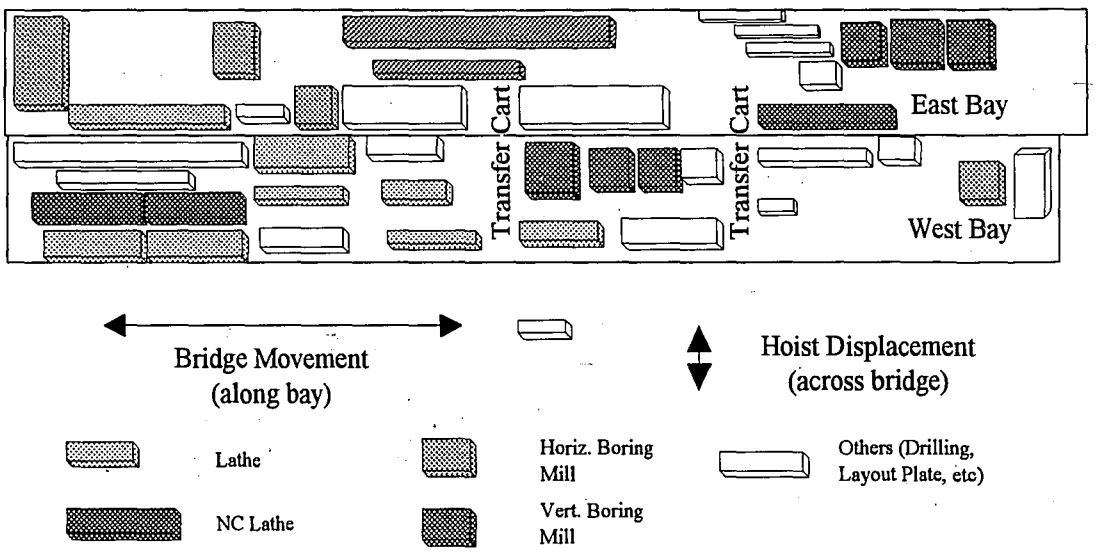


Figure 1.1 Approximate Layout of Machine Shop #1 at BethForge.

The general schedule (and additional shop floor information) dictates the order in which jobs are to be loaded onto each machine. A related decision is how to schedule the moves done by the cranes such that some measure of performance is optimized.

Crane activity is currently planned in a rolling-horizon fashion by the crane foremen. When doing the planning, the foremen considers all "active" jobs (those having

requested a lift already) and looks ahead searching for "jobs likely to became active". While the total machining times are quite variable, forecasting the exact time a job will become active is difficult. However the forecast can be much more precise as jobs near completion on the machines (say within two hours of completion).

Under these conditions, the actual crane scheduling process can be considered static for short periods of time. Rolling horizons and look ahead can be used to take into account the dynamic behavior of the jobs and machines in the longer term.

1.2 Decisions in Constructing Crane Schedules

Upon constructing the shop schedule, there are three sets of decisions to be made for a detailed specification of crane schedules. First, each lift must be assigned to a specific crane. Second, the sequence in which each crane will performs its assigned work. Note that these two decisions are constrained by the potential interference (collision) which may occur between the two cranes. This leads to the third level of decisions: assign priority to the cranes when a collision is detected.

Adding the no-collision (conflict-free) constraint increases the dimension of the problem since now the relative position of the crane must be checked at every moment in time. This issue makes a mathematical programming approach difficult.

1.3 Overview of the Thesis

In what follows, a simplified version of the two-crane scheduling problem faced by the machine shop at BethForge Inc. is studied. The analysis starts by reviewing the main

approaches to crane scheduling found in literature for different shop configurations. In Chapter 3, a formal description of the general crane scheduling problem is presented, with special attention to specify the conditions present at BethForge. Chapter 4 describes the proposed solution approach. Chapter 5 details the experiments conducted to analyze the performance of the heuristic. Finally, Chapter 6 presents conclusions of the research.

Chapter 2 - LITERATURE REVIEW

Literature addressing Crane Scheduling Problems (CSP) is very scare [Matsuo et al. 1991]. The research found in the literature has focused mainly on flow shops with a single crane as found in Printed Circuit Board manufacturing lines (PCB lines) or Assembly of Electronic components to PCBs in CIM environments [Phillips and Unger, 1975][Shapiro and Nuttle, 1988][Matsuo et al., 1989 and 1991][Yih, 1994][Ng 1996][Ge1996]. Under these configurations, the use of cyclic schedules has been shown to produce near optimal solutions by Matsuo et al. [1991].

Although more interesting than the single crane problem, both theoretically and because of the potential for higher system performance, very few references to the two-crane scheduling problem has been found in literature after the work of Lieberman and Turksen [1981 and 1982]. In this chapter, a short description of the approaches described in the literature for the single and two-crane scheduling problem is presented.

2.1 Single Crane Scheduling with Time Window Constraints

The problem in this line of research is to find a schedule of crane movements from one station to another (usually tanks with chemicals), so as to maximize the system throughput. The time a part is allowed to stay in a given station is constrained by minimum and maximum values (time window). Since there is only one crane, interference is not an issue in the problem.

The most common case in this class of problems is the PCB manufacturing line. Usually, a PCB line has 10 to 20 different tanks, plus a load/download station (at one of the extremes of the line). Parts are loaded into carriers and then transported between stations by the crane and left there for a specific range of time (time window). While the carrier with the part remains submerged in the tank, the crane goes to pick up some other carrier to perform a similar operation. All carriers follow an identical sequence of tanks and the number of jobs is considered big enough to assume continuous operation (in a cyclic fashion). Each tank has capacity to hold only one carrier at the time. Some variations from the basic system are: alternative layouts of the line, multiple hoists, duplicated tanks, different load/download configurations [Shapiro and Nuttle 1988].

Phillips and Hunger [1975], proposed a mixed integer program formulated as a one machine problem with sequence dependent setups defined by the time it takes the crane to move from the end locations of its last activity to the beginning location of next job. The proposed model focuses on scheduling the departing times of three fixtures (carriers) from tank to tank. Several groups of constraints ensure the precedence of tasks, time windows for the carriers to stay in a tank, and "left to right" flow of loaded carriers. They proposed a theorem which establishes that the optimal solution of the model leads directly to the solution of the minimum cycle schedule for the crane by simply following the path dictated by the optimal carrier removal times.

Shapiro and Nuttle [1988], consider the same system as Phillips and Hunger. They propose an enumerative algorithm involving the solution of a large sequence of small

linear programs. Their approach is more flexible in that it allows multiple load/download configurations and duplicate tanks. Five test problems are solved optimally, reporting CPU times in the range of 2.75 sec. to 255.51 sec. for problems of 12 processes. The reduction of cycle times over the current solution (line vendor's) best ranges from 2.2% to 10.2%. The number of LP's solved and CPU time required vary widely from problem to problem without any correlation with the number of processes. An additional analysis of an early termination of the optimization is done. If the enumeration stops after the first feasible cycle is found, CPU time for test problems varies from 1.47 sec. to 31.08 sec. with a maximum distance from the optimal solution of 7.5%. The optimal solution is found (at early termination) in two out of five test problems.

Ng [1996] considers the case where the times for inter tank moves are decision variables (different from previous approaches that consider the starting time in each tank as the decision variables) . Under this assumption, the hoist can wait for the next operation after the drip-off of an intertank move. In other words, it is allowed to use the hoist as a "buffer" other than just a transporter. He proposes a mixed integer formulation solved by an efficient branch and bound algorithm. Computational results of randomly generated problems show that, as the number of tanks in the line increases, both the average computing time required to solve the problem and the percentage of computing time spent on finding the first solution also increase. However, the percentage of computing time spent on solving linear programs decreases as problem size increases.

Matsuo et al. [1989 and 1991], address the dry end of electronic manufacturing. Here a crane moves magazines with printed circuit boards between workstations in a CIM environment. Boards contain paste that adheres to the components attached at the workstations. The paste is usable for up to eight hours after which it dries and the boards must be scrapped. Other than this, no other time constraints exist in the workstations. They proposed a heuristic to compute cyclic schedules and show the near optimal nature of such solutions. They show that cyclic schedules are superior to dispatching rules in a repetitive and deterministic manufacturing environment. The basic heuristic consist of the crane moving cyclically every T units of time. Upon arrival to station i, the crane will move the existing part to station $i+1$ if it is ready to be moved, otherwise the crane will continue traveling empty to station $i+1$. This procedure is extended to consider multiple product types and parallel flexible machines. A heuristic procedure is used to generate different sequences of products to be included in a cycle. Then a maximum cost circular flow formulation is used to calculate the minimum cycle time for that sequence. The heuristic is tested against simple dispatching rules for assigning jobs to the crane. One test configuration considering 4 types of products was tested. The proposed heuristic shows a much higher throughput rate and lower WIP than the compared rules.

Yih [1994] studies the general flowshop (non-cyclic) problem, where different products arrive randomly to the line. Although the jobs are allowed to have different routings and skip tanks, the constraint of unidirectional flow (left to right) is preserved. She proposes a two phase algorithm for deciding when to enter each job to

the line and assign them to the corresponding workstations. Upon arrival of a job to the system, the algorithm starts by assigning the current time as initial entry time. Following, a two-phase procedure computes the right entry times to achieve feasibility in the schedule. Phase one considers the conflicts among jobs requesting the same machine (tank), while phase two deals with the availability and schedule of a single hoist. The algorithm is based on the concept of tolerances of processing times (time windows) instead of minimum processing time as the driver of the scheduling procedure. If a hoist conflict is detected in phase two, the procedure delays the entry of the job to the line, or extends the actual processing time in a tank. In the last case, current processing is extended and the starting times of following tanks are also delayed. The study uses the completion time of the first 100 jobs completed as the performance measure. The performance of the procedure is established by comparing the results against a basic algorithm which requires exactly the minimum processing time on each tank (zero-tolerance). A simulation study is performed to analyze the procedure under different distributions of processing and transportation times. The study shows that the procedure based on tolerances outperforms the basic procedure. In addition, there are significant interactions between hoist operating speed and mean and variance of processing times (both individually).

Ge [1996] and Ge and Yih [1995] suggest an approach that combines real-time scheduling and a heuristic procedure. An incomplete branch and bound procedure is employed to analyze the results of choosing different jobs when the crane becomes available. A depth-first search technique is used. The priority of selection of "next job

to move" is higher for new jobs waiting to be loaded to the line. Next priorities are assigned in a longest processing time basis. Conditions for feasibility of each node are proposed and used to reduce the depth of branches. The method can handle multi-type job scheduling and does not result in the production of defective jobs as many of the method previously listed.

2.2 Two-Crane Scheduling in Flow Shop

In this line of research, the main issue is the existence of interference between the two cranes sharing a common track. As before, parts are restricted to follow a unidirectional (left to right) flow.

Lei and Wang [1991] present a heuristic algorithm to find schedules in PCB lines with two cranes. They propose partitioning the system in two mutually exclusive sets of workstations. Each crane is then assigned to serve exclusively the workstations in one set. A sequence of partitions is examined, where each partition is mapped to a one-crane problem. Cycle times are computed independently for each crane looking for the common-length cycle that maximizes throughput.

Optimal cycles are obtained for small problems and compared to the results from the heuristic. They found that the relative deviation from optimal cycle (in terms of percentage) decreases as the number of stations (and consequently the number of crane operations) increases, while the absolute deviation increases too.

Armstrong et al. [1996] consider the problem of minimizing the number of transporters in a cyclic processing line under time window constraints. To avoid traffic

collisions, a partition of operations into groups is employed. Each group is served by an individual crane. A search method is proposed to maximize the size of the groups and consequently minimize the number of transporters. The method exploits the fact that the maximization of size groups produce dual programs specially structured as shortest-path problems.

2.3 Two-Crane Scheduling in Job Shop

In this case of problems, as the constraint of unidirectional flow of jobs is dropped, interference among cranes becomes more important.

Lieberman and Turksen [1981 and 1982] presented a series of algorithms for single track cranes. Although they do not require unidirectional flow of jobs, their algorithms assume independent moves, instantaneous movements of cranes and indefinitely wait for the pieces. One of the algorithms proposed divides the moves into batches of moves non-overlapping in the shop and then assigns cranes to one batch at each time. The algorithm establishes conditions under which interference-free schedules can be found. In the majority of cases they consider identical processing times for the crane operations. In addition, their procedures require that all the moves considered during the planning horizon must be independent.

Our research differs from the basic model considered by Lieberman and Turksen in which, as detailed in next Chapter, we consider a general Job Shop structure where moves are not independent. In addition, we model the movements of the cranes as

non-instantaneous and allow for multiple machines to be located in the same location along the track.

Chapter 3 - PROBLEM DESCRIPTION

The problem considered here is a simplification of the problem faced by the #1 Machine Shop at BethForge as described in Chapter 1. For simplicity, only one bay with two cranes will be modeled. Through the analysis, the layout of the West Bay is used as the base configuration. In this chapter, the model used in the study as well as notation and concepts are defined in detail.

A model of the components is listed on Table 3.1 . The name *move* is employed to refer to crane activities, while *job* is employed on referring to a set of machining *operations* performed on a product.

Table 3.1 Components of the Multiple Crane Scheduling Problem

$\mathcal{C}$: set of cranes $\{c_r \mid r=1,...,m\}$

$\mathcal{L}$: set of job locations $\{\ell \mid \ell = 1,2,...,L\}$

$\mathcal{S}$: set of feasible sites along the common track $\{ s \mid s=1,...,S \}$

M : set of moves $\{m_i \mid m_i = (m_{ia}, m_{il}, m_{it}, m_{id}),\ i=1,2,...,n\}$

$\mathcal{R}$: set of ready times $\{r_i \mid i=1,..., n\}$

$\mathcal{T}$: set of lift/download times $\{ (t_{i1}, t_{i2}) \mid t_{i1} \geq 0, t_{i2} \geq 0 ,\ i=1,2,...,n\}$

$\mathcal{J}$: Set of jobs $\{ j \mid j=1,..,J \}$

O : Set of machining operations $\{o_{jk} \mid j=1..J,\ k=1..K \}$

The common track contains S *sites* where *resources* are located (machines, buffers, etc.). Each *job* j consists of a series of k *operations* O_{jk} performed on the different resources of the shop. A *move* is required any time an operation is finished, or to load the product onto a machine. Each move is decomposed into four basic activities: m_{ia},

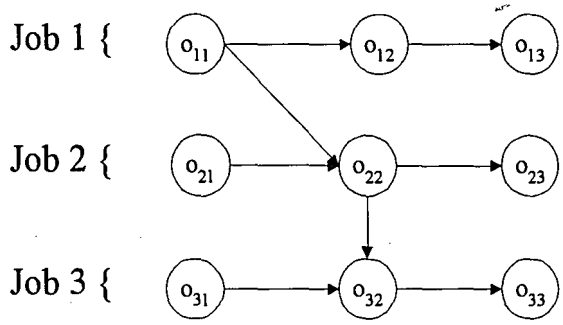
approach the lift location ℓ_{i1} (also called dead head trip); m_{il}, *lift* or *pick-up* the part (includes the setup of lifting accessories and the actual lift operation); m_{it}, *transport* the part to the location ℓ_{i2}; and finally m_{id}, *download* or *drop-off* the part (includes set up at the receiving resource and removal of lifting accessories). In some special cases such as emptying chip boxes, moves consist of an additional intermediate operation (such as turning over, or emptying the box) before being transported to the download location (the same lift location). However, since these moves have a low frequency and can be delayed during low utilization periods, they are not considered in this research.

The specification of the locations and processing times generates two main quantities for a move, its *duration* and *length.* Duration is measured as the total time that a crane spends lifting and downloading the part plus the known transportation time (approach time is not included because it depends on the previous move in the sequence). On the other hand, *length* of a move is measured in units of length or number of sites between lift and drop-off locations. *Work content* is the summation of the duration of all the jobs in a given group. The *centroid* of a move is defined as the weighted average location of the known segments of the move.

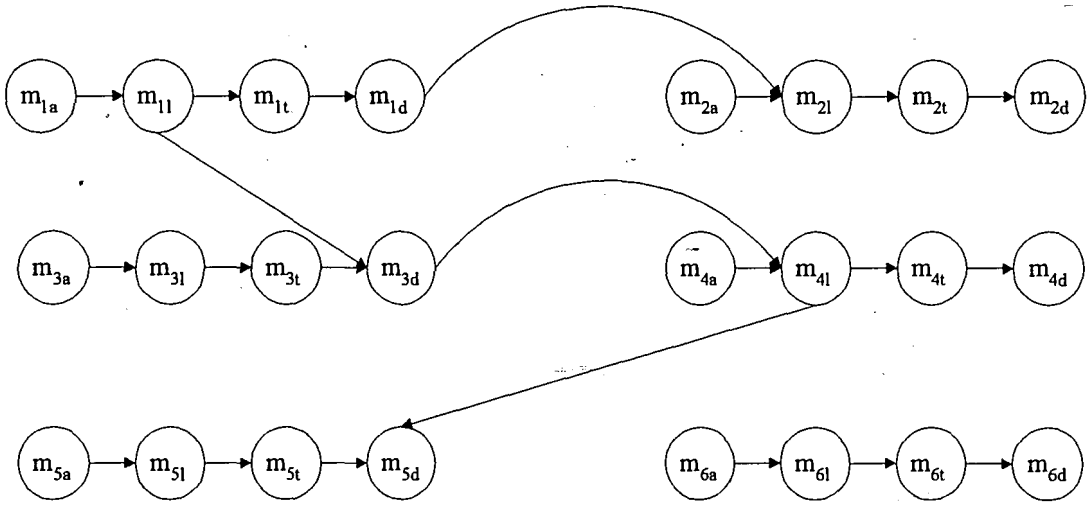
The shop schedule generates a precedence graph P_{shop} as the one shown in figure 3-1 where each node represents a machining operation and each arc represents a precedence relation among them. This graph is the base for constructing the move precedence graph P_{moves} used later for conflict handling. In P_{moves}, each node

represents a basic component of a move while arcs represent precedence relations among the components.

Graphs P_{shop} and P_{moves} are related in the sense that the transition between two consecutive operations in P_{shop} requires a move represented in P_{moves}. In other words, for each pair of consecutive operations of a job there is a required move.



a)



b)

Figure 3-1 Example of precedence graphs a) P_{shop} and b) P_{moves}

The sequence of space-time tuples that a crane visits when moving from one location to another will be called a *crane tour*. A *feasible crane tour* (or *schedule*) contains

only tuples which do not overlap. Figure 3.2 shows a crane schedule as derived from the Gantt chart originated by the shop scheduler. The plot on the bottom of the figure depicts two crane tours in a *time-way diagram* as shown in Shapiro and Nuttle [1988]. In the diagram, time is represented on the X-axis while the Y-axis represents locations in the track used by the cranes. A tour is represented as a succession of connected segments. Horizontal segments correspond to periods of time when that crane is not moving either because of a lift or download activity, or because it is idle. A diagonal line represents the movements of the crane from one location to another. A very useful feature of time way diagrams is that magnitudes in the Y-axis are proportional to actual distances in the shop. The slope of a diagonal corresponds to the average crane speed. Note that in this example, the cranes can be scheduled without affecting the times suggested by the Gantt chart. In general, this is not expected to be the case. In this problem, a buffer location (L1) exists to avoid *machine-deadlock* in the shop. This is consistent with practice at the #1 machine shop.

A *collision* occurs whenever one of the cranes tries to access a segment of the track being occupied by the other crane. To avoid collisions, the activity of each crane must be scheduled considering the current and future status of the other crane and vice versa. A *feasible schedule* can be seen as a time-way diagram where the lines defining each crane tour never intersect.

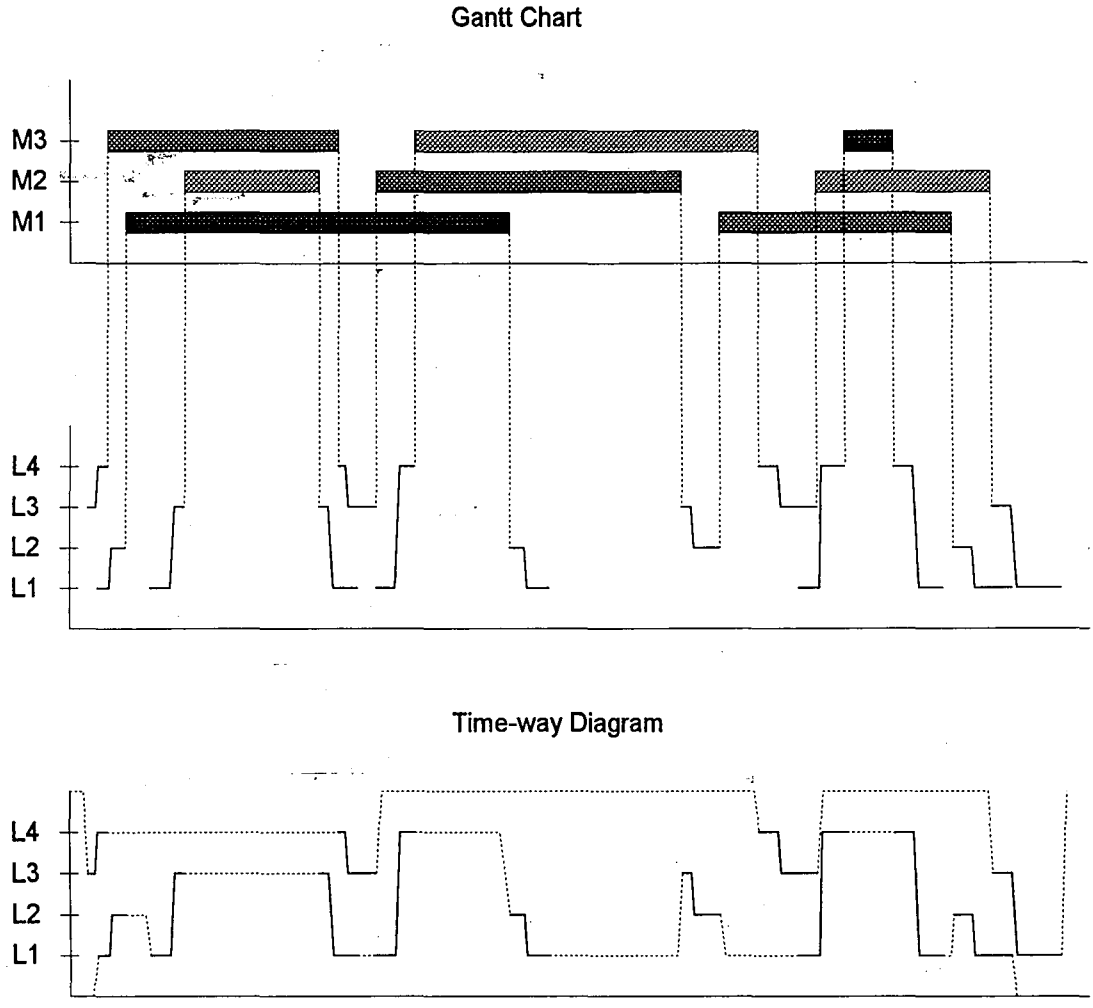


Figure 3-2 Gantt chart and Time way diagram for a feasible two-crane schedule

3.1 Precedence Relations among moves

The resolution of the shop scheduling problem provides two types of precedence relations among the moves in the CSP. The first relates moves corresponding to consecutive operations of a job. Here the precedence establishes that the lift task of a successor move cannot be done until the download task (and eventually the required machining) of the predecessor move has been completed.

The second set of precedences arises from moves associated with machining operations done at a common machine. As mentioned before, the shop schedule dictates the sequence of jobs in the conflicting machine. The precedence relation now establishes that the download task of a successor move must follow the lift of the predecessor move on the shared machine.

In addition to these relations imposed by the shop schedule, two new types of precedences appear during the resolution of the CSP. Precedences derived from the crane sequences relate two consecutive moves done by a single crane. The approaching task of a move must be preceded by the download of the predecessor move in the crane sequence.

Finally, during the resolution of conflicts, the assignment of priority to one crane produces a precedence relation between the two conflicting moves.

Figure 3-3 depicts the different precedence relations that appear in the CSP.

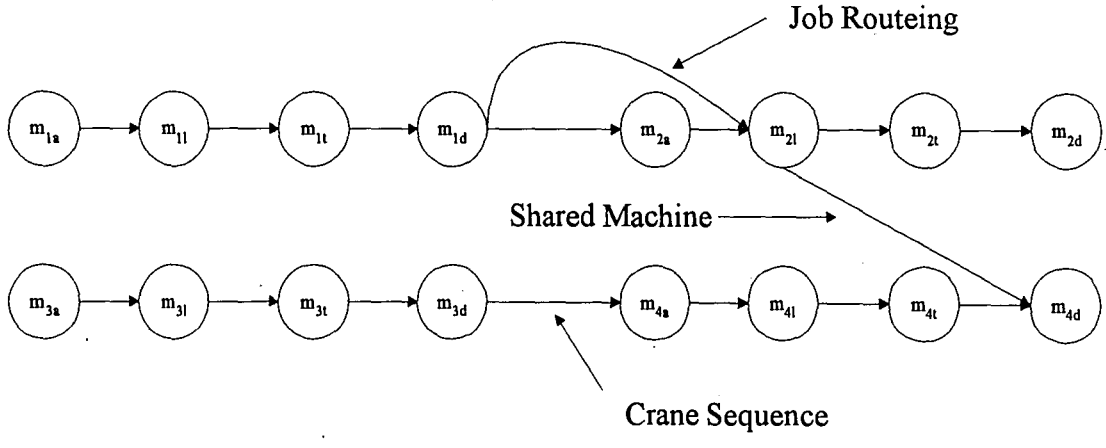


Figure 3-3 Precedence relations

3.2 Problem assumptions

On specifying the exact configuration under study, the assumptions and system simplifications made are similar to those from Lieberman and Turksen [1981], except for the assumption of instantaneous crane displacements. We model crane travel time as well. The detailed list of assumptions is:

(1) The number of moves is finite.

(2) The ready times are known.

(3) The set of move times (lift and drop-off) is deterministic and known.

(4) The cranes are identical.

(5) Crane travel between two locations is not instantaneous. Crane speed is the same when traveling empty or loaded and is homogeneous along the track.

(6) There is no pre-emption of moves or jobs.

(7) Each crane can process at most one location at a time.

(8) Each move is processed by exactly one crane.

(9) The two cranes cannot access the same location at the same time.

(10) The common track is divided into equidistant sites where the resources (machines and buffers) are located. These sites are the only locations where a crane can stop.

(11) Initial and final locations of the cranes are the same and pre-specified (usually the two extremes of the bay).

(12) Cranes and machines never break down.

Assumption (6) means that once a *move* is initiated, it is performed without interruption until it finishes. This is specially important in the #1 Machine Shop

because, due to safety considerations, a crane may never wait loaded for the track to became empty. Thus, all conflict resolution measures must be taken before a crane starts a move.

Figure 3-4 shows the basic configuration used in the CSP considering the common track having *s* feasible sites. Sites *0* and *s+1* are the extreme locations of the bay and can be accessed by the corresponding crane only. The s "internal" sites host the resources of the shop (several resources may occupy the same site on the Y-axis). Assumption (10) will eventually produce sites with no resources allocated.

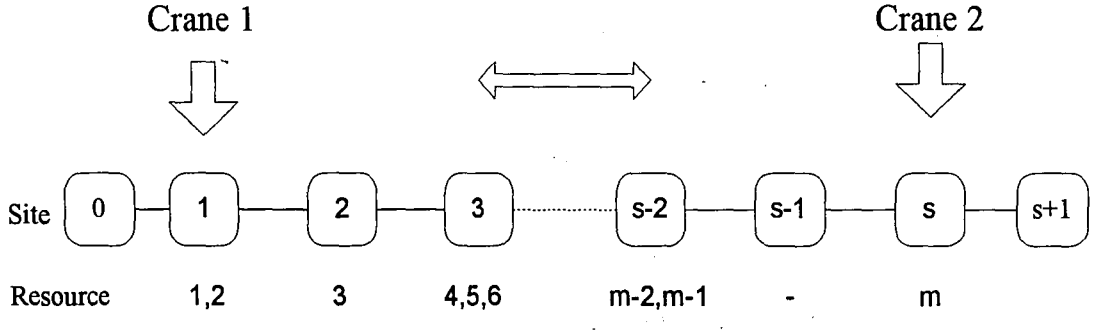


Figure 3-4 Simplified configuration of a two-crane bay with multiple resources per site

When constructing the schedule, there are two additional activities that a crane can be forced to perform due to interference with the other crane:

i) a crane may *wait idle* until the route needed to perform its task becomes free. In this case the waiting crane will stay in its current location until the other crane gets out of the way of its route. This does not mean however that the waiting crane needs to wait

until its route becomes free before starting its operation, but rather the time necessary to avoid a collision.

ii) a crane must *move out of the route* of the other. In this case, the current crane is idle and blocking the route of the other crane that is already performing a task. The second crane may even be traveling empty, but has already started the approach task of a move.

Due to its simplicity and ease of interpretation, it is assumed that the objective function of the CSP is to minimize the *completion time of all the moves* (crane makespan). Note that this measure may be different from the objective criteria used for scheduling the machining operations in the shop. In this case, minimizing the makespan of the cranes allows us to determine if a given assignment of moves can be performed during a given time span (say a shift or fraction of it). A different measure of performance that could be used is the average waiting time for the parts. In that case the wait time is considered from the moment a request for lift is made until when the crane actually picks-up the part.

3.3 Problem difficulty

The general crane scheduling problem has been shown to be NP-Complete [Lieberman and Turksen 1981], even for the simple case of a one hoist single-product flowshop [Lei and Wang 1991]. Moreover, relaxing some of the hard constraints allows us to relate the CSP with other combinatorial problems known to be extremely hard. For example, if the interference between the two cranes is ignored and all the moves are

independent, the problem can be reduced to scheduling identical parallel machines with sequence dependent setup (the approaching times correspond to the setup times and the summation of the lift, transport and download times corresponds to the "processing" time). If, in addition, the approaching of a crane to the lift location is ignored, the problem reduces to scheduling parallel processors without setup times problem.

The complexity of the problem can be determined easily for the case of all-independent moves. For a given set of n moves there are $n!$ different ways of sequencing the moves (independent of the assignment to cranes). For each sequence there are $(n-1)$ ways of assigning them to non-empty crane sequences. So, there are $(n-1)\ n!$ different ways to construct the crane sequences. In addition, in the worst case, a move will produce collisions with all the other moves in the other crane. That means 2^{n-1} different solutions generated by assigning priority to one crane or the other at each collision point. Finally, in the worst case there are $O((n-1)\ n!\ 2^{n-1})$ different solutions for the all-independent moves two-crane scheduling problem. If precedence relations are included in the formulation, the number of feasible sequences is reduced but it is still a fraction of $n!$. On the other hand, the number of priority assignment is still 2^{n-1} in the worst case.

The degree of difficulty of the problem leads us to develop a heuristic procedure as described in detail in the following chapter.

3.4 Prevention of Machine Deadlocks

Deadlocking is a critical scheduling and control problem, especially in an automated manufacturing system. A manufacturing system is said to be in deadlock when "parts are assigned to machines such that further flow of these parts is permanently inhibited" [Ramaswamy et al. 1996].

Wysk et al. [1991, 1994] propose a graph-theoretic approach for on-line deadlock detection and resolution. The method is based on detection of "circular waits" as circuits in the graph. The procedure is called each time a new part arrives or a move is requested. The recovery procedure suggested randomly chooses one of the deadlocked parts and transfers it to a specially reserved storage. The rest of the deadlocked parts are then transferred sequentially to the next destination on their routings [Wysk et al. 1994].

Ramaswamy et al. [1996] proposes mathematical formulations for a-priori deadlock avoidance. The models are developed for workstations with no buffer and finite capacity buffers. Due to solution time considerations, they suggest some heuristic procedures to be used as a solution strategy.

Recognizing the importance and difficulty of the problem, but considering it out of the reach of this research, a simpler approach has been used to deal with machine deadlocks.

The shop schedule is given, and is used to construct the graph of move precedences. Then, circular references are detected by inspection. In cases where cycles are present, new moves are included in the problem to "break" the cycle. As in Wysk et al.

[1994], one of the parts generating the deadlock is moved to a buffer, then the others proceed regularly. At the end of the deadlock recovery procedure, the selected part is moved to its original target destination. Consequently, the updated precedence graph does not contain the circular reference as before. The idea is illustrated on Figure 3-5.

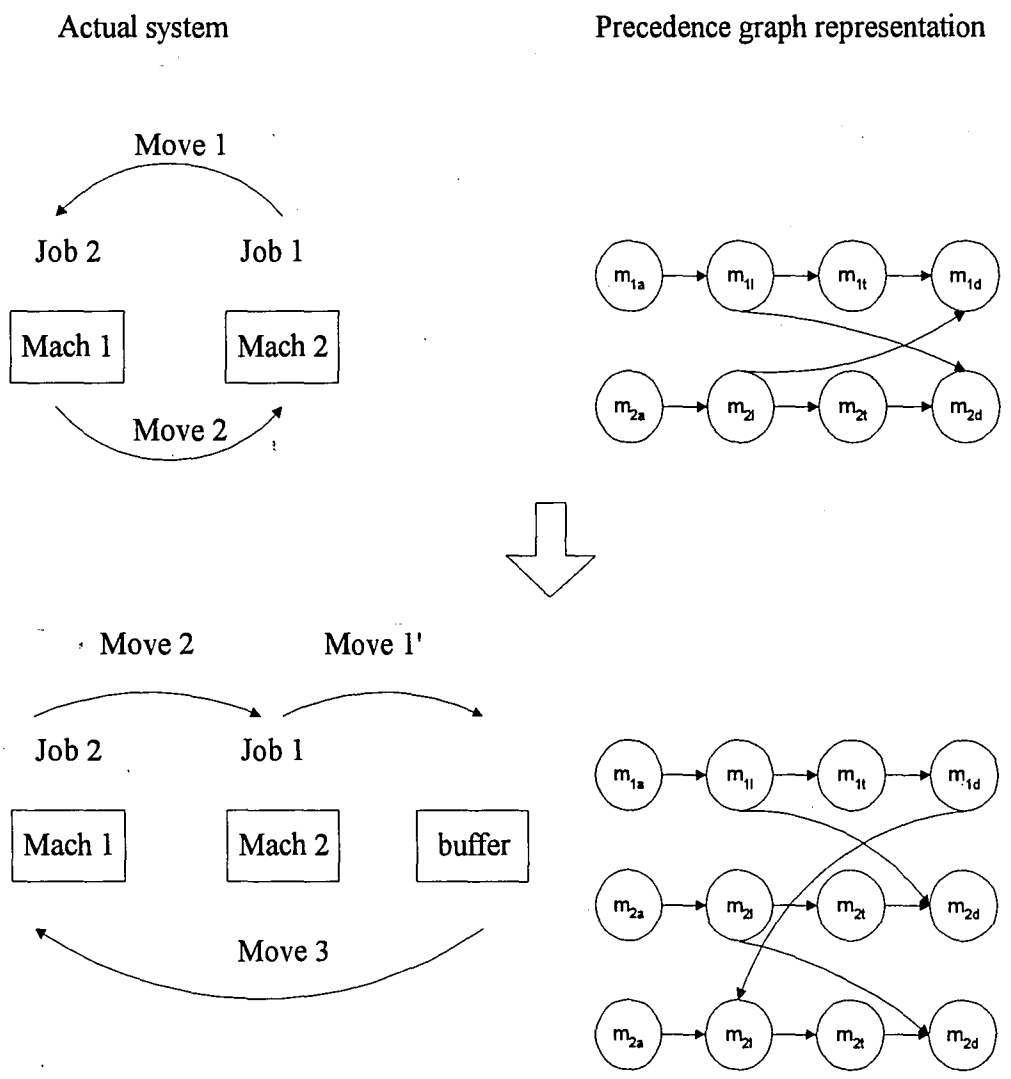


Figure 3-5 Procedure for Machine Deadlock Recovery

In the Figure, job 1 needs to be moved to machine 1 which is currently loaded with job 2. A deadlock occurs because job 2 needs to be moved to machine 2, being used by

job 1. This is represented in the actual system diagram (top left) and the existence of a circular reference (cycle) in the precedence graph P_{moves} (top right).

The diagram at the bottom left shows the inclusion of a buffer to break the deadlock. Job 1 is moved to the buffer leaving machine 2 available to receive job 2. After job 2 is moved to machine 2, job 1 can be removed from the buffer and loaded to machine 1. Note that the new precedence graph does not have any cycle (bottom right).

Chapter 4 - HEURISTIC DESCRIPTION

In this chapter, the foundations for the proposed heuristic are detailed. As described before, there are three decisions to make in obtaining a feasible schedule for the cranes:

i) Assign each move to a specific crane

ii) Decide the sequence of moves for each crane

iii) Assign priority to one of the cranes in case of collision

These decisions do not need to be made in this exact sequence, however they are strongly interdependent. For example, the assignment of two moves to different cranes may or may not produce a collision depending on the position of the moves in each crane tour.

In an attempt of exploiting the deterministic nature of the problem under study, a search-based approach is proposed that divides the problem into two sequential stages as shown in Figure 4-1. The first deals with the assignment of cranes and sequencing of moves, while the second stage handles the detection and resolution of collisions. This way the problem can be divided into two separate problems that are easier to solve using customized procedures. Since the two problems are not independent , an iterative procedure is required to search for the global optima or a good sub-optimal solution.

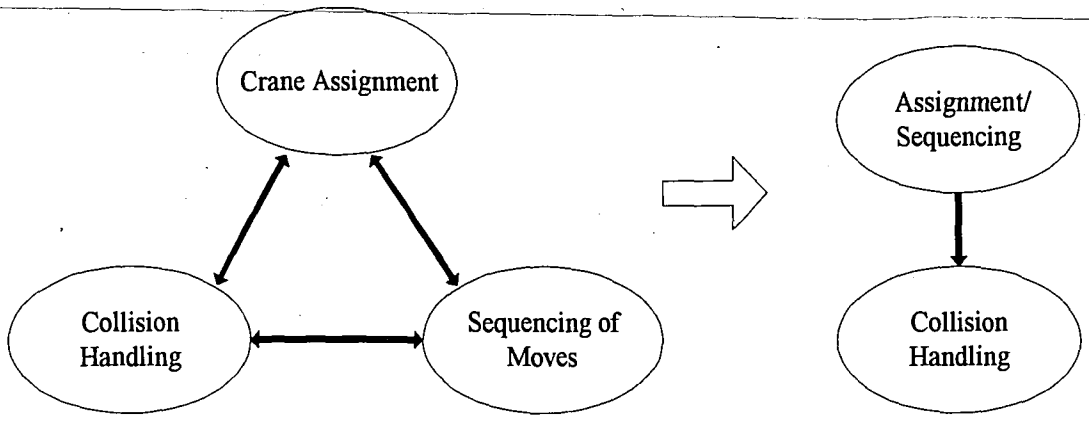


Figure 4-1 Decomposition of CSP into sequential decisions

The main advantage of this decomposition is that simple or well known procedures can be used to solve each subproblem. For example, the assignment/sequencing can use any dispatching rule-like procedure as far as the solution generated satisfies the precedence relations dictated by the graph P_{moves} .

4.1 General Approach

Figure 4-2 illustrates the general approach suggested. On it, a new deadlock-free candidate solution S_{df} is generated at each iteration. Since this solution has been generated without considering the crane interference constraints, it may still be "crane-infeasible". If the solution presents a makespan shorter than the best found so far, it is called a *good candidate*. Good candidates are analyzed in detail by resolving crane collisions in the module "CheckBest". This is a branch and bound based procedure that checks the solutions resulting from different assignments of priorities at each collision point. For the given sequencing/assignment solution, the procedure will find the best

collision-free solution. The search continues until a pre-specified finishing condition is reached.

In what follows, the two procedures as well as the search strategy employed are fully explained.

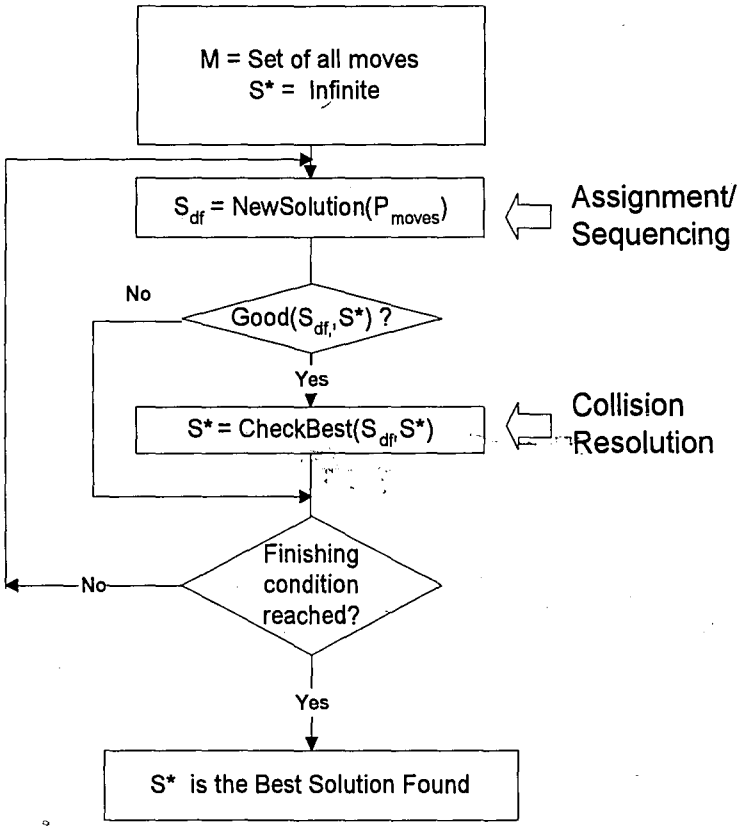


Figure 4-2 General approach of the proposed heuristic

4.2 Solution of Assignment/Sequencing Sub-problem

The objective of this stage is to obtain an assignment/sequencing solution for the cranes which is feasible for the given shop schedule. This solution indicates the order in which each crane will perform the moves assigned without producing a deadlock of

the shop. In order to obtain such a solution, the assignment/sequencing procedure relies strongly on the graph P_{moves} as described on Figure 4-3.

The procedure FindAssignmentSequencing make use of some properties from the solution generated at this stage which are conjectured to have a positive impact on the minimization of the makespan of the cranes. They are:

i) Balanced work load on both cranes to avoid excessive idle time on one of them.

ii) Assignment of moves according to zoning criteria to minimize the possible collision between the cranes derived from one of them working near the opposite extreme of the bay.

iii) Sequencing of moves according to their ready times.

The procedure starts by checking that no circular precedences (cycle) exists in the graph P_{moves} (which will result in a deadlock in the cranes as explained in Section 3-4). If no such cycle is present, the list which will contain the sequences of moves for the cranes (*Seq[1]* and *Seq[2]*) are empty, all the moves are unmarked, and the counter of number of moves assigned (*MovesAssigned*) is set to zero.

In step 1, the centroids (*Centroid[i]*) for each move are computed as well as the overall weighted centroid (*OC*). Cranes are assigned to each move by comparing the centroid to *OC* . This criteria attempts to establish some zoning in the assignment while balancing the workload. If the moves are sorted by their centroids, the overall weighted centroid represents the point which divides them into two sets having the more balanced workload. Transferring one of the closest moves to OC from one set to

the other will increase the unbalance in the assignment. Additionally, the ready times for each move are computed based on the precedences described in ρ_{moves}.

Step 0. AssignedMoves = 0
Unmark all the Nmoves
Empty lists Seq[1] and Seq[2]
If graph ρ_{moves} has no directed cycles, goto 1
otherwise, return error, the system contains a deadlock derived from the precedence relations between the operations. Exit

Step 1. For each move *i* compute its centroid. Compute the weighted overall centroid as:

$$OC = \frac{\sum_{i=1}^{NMoves} Centroid[i] * Duration[i]}{\sum_{i=1}^{NMoves} Duration[i]}$$

Assign each move *i* to crane *k* such that:

$$k = \begin{cases} 1 & \text{if } Centroid[i] < OC \\ 2 & \text{otherwise} \end{cases}$$

Compute the ready time for the lift task of each move

Step 2. Find all the moves in ρ_{moves} having no predecessors or those which all their predecessors have been assigned already. Call these moves *Candidates*.

Step 3. From the candidate moves, chose the one with smallest ready time

Step 4. Add move *i* at the end of the sequence of moves Seq[k] for crane *k*.
Mark move *i* as assigned.
Increment AssignedMoves by 1.

Step 5. If AssignedMoves < Nmoves, goto 2
otherwise the sequences for each crane are Seq[1] and Seq[2], respectively

Figure 4-3 Procedure to find Assignment/Sequencing solution

In step 2 and 3, the candidate move with the smallest ready time, say move *i*, is selected. Ties are broken randomly.

In step 4, the selected move is append to the corresponding crane assigned in step 1. The move is labeled as "assigned", removed from the list of candidates, and the counter *MovesAssigned* is incremented by one. The list of candidates is updated with the moves having move *i* as predecessor and not having an unmarked predecessor.

In step 5, the termination condition for the assignment/sequencing procedure is checked. If all the moves (*NMoves*) have been assigned, *Seq[1]* and *Seq[2]* contains the resulting sequences. Otherwise, goto step 3 to assign the next move.

Termination is guarantied by checking for the existence of cycles in step 1.

4.3 Solution of the Collision Handling Sub-problem

Once the assignment/sequencing problem has been solved, Seq[1] and Seq[2] contain the sequence of moves assigned to cranes 1 and 2, respectively. These sequences are feasible for the machines in the shop (do not contain deadlocks) but may still produce collisions between the cranes as the moves are performed in time. Conflict handling consists of two basic functions: conflict detection, and conflict resolution. These functions constitute the core of a collision handling procedure which iterates on the tours until the conflicts have been fully resolved. Both functions are explained in the following two sections.

4.3.1 Conflict Detection

As mentioned in Chapter 3, a conflict can be seen as crossing (or overlapping) segments in the time-way diagram. Using this basic idea, a scanning procedure follows the crane tours defined in the diagram until a crossing is found or until reaching the end of the tours. Because of the linear characteristics of the crane movements (start-stop dynamics are ignored), only the extreme points of each segment need to be stored

to define the tours. Consequently, the problem faced here consists on determining the existence of intersection for a given pair of segments which belong to different cranes. Obviously not all the possible combination of segments need to be tested, but only those being simultaneous.

Once a conflict has been detected, the conflict resolution function is called to generate new tours which are expected not to contain the last conflict but still need to be scanned for the succeeding tasks.

4.3.2 Conflict Resolution

The resolution of a conflict consists on taking the corrective measures such that the operation of the cranes can be continued. The main strategy for conflict resolution is the insertion of idle time in one of the tours to delay its entrance to the conflicting locations. The crane assigned high priority can perform the assigned move in a "free track" while the other waits idle until its required segments of the track become available. Recall from assumption (6) in Chapter 3, that idle time can be inserted only before a move actually starts. This imposes an additional constraint in the start and finishing times of the tasks in a given move.

Figure 4-4 shows an example of conflict resolution after a collision has been detected between the approaching task of move 3 and the approaching task of move 4. Note that in this case the priority is assigned to crane 1 (doing move 4) forcing crane 2 to go outside the region needed by crane 1. Crane 2 waits idle until it can access its required zone safely.

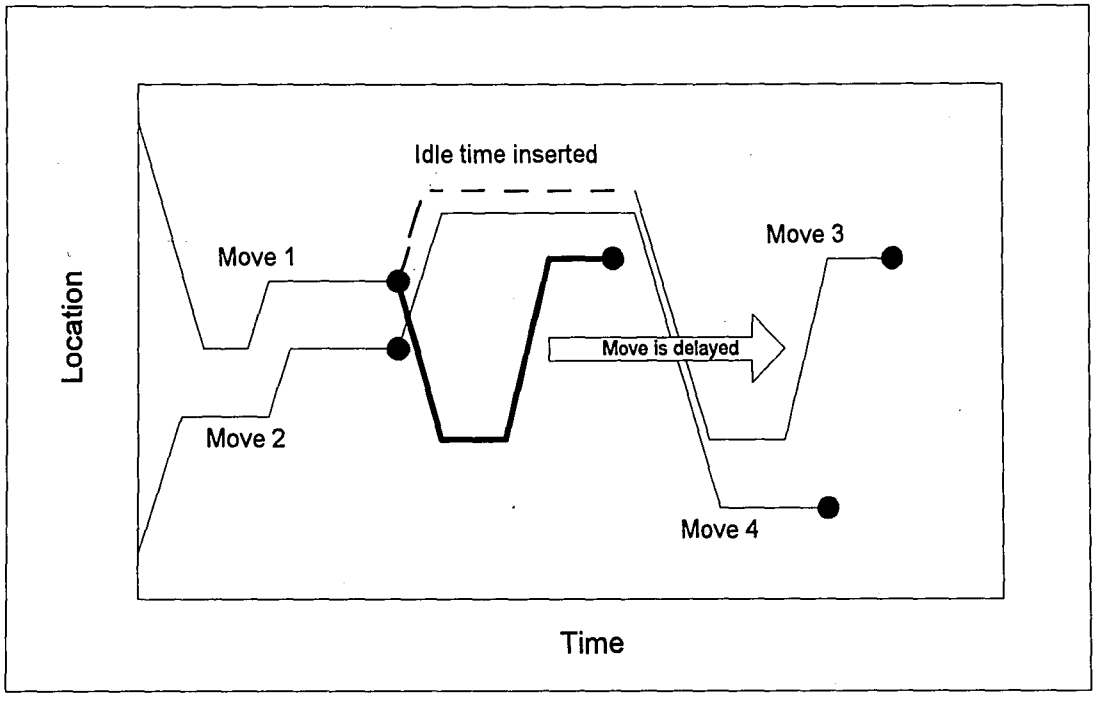


Figure 4-4 Example of conflict detection and resolution

The detailed pseudo-code definition of the collision handling procedure is listed in Figure 4-5. The main variables used by the procedure are: *BestFound*, containing the best objective value found at any point; *Cr*, array containing pointers to the segments currently being tested for collision; *L*, the current depth into the branch and bound tree (it corresponds to the number of different pairs of moves that have collided so far). The names in italics are functions and procedures described in the following paragraphs.

The procedure is based on an external REPEAT cycle which will terminate when no more levels of the branch and bound tree are left to inspect. A nested WHILE loop implements the scanning of the tours, looking for collision points. When a new collision is detected, the procedure generates two new branches of the tree. One of the branches is chosen for inspection and the status of the current node is saved to be

recovered during the backtracking operation. The function *ResolveCollision* determines the feasibility of the resulting partial crane schedules. Then, the procedure check for the reaching of a bound condition to decide if it needs to backtrack the tree or to continue the scanning of the feasible partial solution.

```
Best Found <- BestObjectiveValue
ResetMarksOnMoves
FOR i = 1 TO 2 DO Cr[i] <- Seq[i]
L <- 0
REPEAT
{  ExitCurrentBranch <- FALSE
   WHILE NOT(EndOfToursReached) DO
  {  IF DetectCollision(Cr) THEN
     {  IF NewCollision THEN
        {  L <- L+1

           IF (Branch(ExitCurrentBranch)) THEN SaveTours
        }
        IF ExitCurrentBranch OR NOT(ResolveCollision(Cr))
        THEN GoTo EndBRanch
        IF (ComputeObjectiveValue(CraneTours)<BestFound) THEN
        {  ExitCurrentBranch <- TRUE
           GoTo EndBranch
        }
     }
     ELSE
          AdvanceScanning(Cr)
  }
  EndBranch:
  IF  NOT(ExitCurrentBranch)
      AND (ComputeObjectiveValue(CraneTours)<BestFound) THEN
      UpdateBestSolution
   BacktrackTheTree;
}
UNTIL (L=0)
CheckBest <- BestFound
```

Figure 4-5 Pseudo-code for CollisionHandling Procedure

Each time the procedure gets out of the internal WHILE loop, means that a branch has reached its end and it is time to go back one level. The exit condition for the REPEAT loop forces the procedure to complete the enumeration at all the levels of the tree.

4.4 Searching Procedure

Storer et al. [1992] have shown the usefulness of search spaces defined by perturbing the data of a problem.

The heuristic makes use of this idea by promoting a perturbation scheme during the solution of the assignment/sequencing part of the problem. As shown in Figure 4-2, a new solution is generated at each iteration of the procedure in the module Assignment/Sequencing. This new solution is generated by perturbing the computation of the ready times and centroids of the moves (as described in Figure 4-7). Then, the procedure described in Figure 4-3 is applied to the perturbed problem to find the assignment/sequencing solution for current iteration. If the solution generated this way satisfies an acceptance criteria, it is accepted as a candidate and passed to the second module of collision resolution as explained in Section 3.3.

Uniformly distributed random variables R1 and R2 are used to perturb the original data of the problem as described on Figure 4-5. The perturbations are defined as factors of problem parameters . The specific values of the factors Weight1 and Weight2 are found during the tuning of the parameters.

```
Using the data from the problem, compute:
        MaxPerturbR2 = Max { 1, (MaxReadyTime,MinReadyTime) }
        MaxPerturbR2 = Max { Centroid[i, i=1,NMoves }

At each iteration :
        Generate random value r1 from distribution U[-MaxPerturbR1,MaxPerturbR1]
        Generate random value r2 from distribution U[-MaxPerturbR2,MaxPerturbR2]
        FOR i=1 TO NMoves DO
        { NewMove[i].R = Move[i].R + r1 * Weight1
          NewCentroid[i] = Centroid[i] + r2 * Weight2
        }
```

Figure 4-5 Perturbation Scheme used to generate new candidate solutions

The procedure iterates until the finishing condition is reached. For the test runs, the termination criteria was established as having generated a large enough number of solutions that no major improvements can be achieved.

The next chapter gives results obtained from a large number of test problems generated for four different system configurations.

Chapter 5 - COMPUTATIONAL RESULTS

A set of experiments has been designed to study the performance of the heuristic under different type of problems. The experiments are divided into four parts:

i). Static problems consisting of all independent moves. That is, at any moment in time, the cranes can perform any one of the moves because they are independent and active at time zero.

ii). Dynamic problems consisting of all independent moves. Same as before, but now the ready times of the moves are known random variables.

iii). Job shop with short machining times. Under this configuration, there are precedence relations between moves when they are associated with different operations of a same job. The ready time of a move is a function of the ready time of its predecessor moves and the availability of the target machine. In this case, machining times are set roughly equal to the move times.

iv). Job shop with long machining times. Same as (iii), but machining times are set much greater than the transportation times.

Specific questions the experiments set out to answer are as follows: what is the best parameter setting in each case considered? is there a universally sound parameter setting?, does the heuristic provide better results than a computationally simple approach?, and how is the quality the heuristic compared to benchmarks?

In what follows, the details of the experiments are explained followed by individual analysis of each case.

5.1 General strategy for generation of test problems

In each of the cases, problems of different size were randomly generated . The basic layout utilized corresponds to the layout of the machines in the West Bay of Machine Shop #1 at BethForge as shown in Figure 1-1. The track was divided into 20 equidistant sites plus an extreme resting location for crane 1. The assignment of machines to sites is shown in Table 5-1. The traveling time between two adjacent sites was estimated as 0.2 minutes.

Site	Machine or Buffer Number
0	none
1	1, and arrival/delivery of parts
2	none
3	18 and 23 (buffer)
4	24 (buffer)
5	19 and 20
6	transfer cart-south
7	22 and 25
8	2 and 21
9	3
10	5 and 4
11	transfer cart-north
12	26 (buffer)
13	6, 7, and 8
14	none
15	9, 10, and 11
16	none
17	12 and 13
18	none
19	14, 15, 16, and 17
20	none

Table 5-1 Resources (machine/buffers) allocated to each site

Special attention has been placed on characterizing the difficulty of a problem.

After several runs of preliminary problems, it was found that the difficulty of a problem increases as the number of potential conflicts increases. Two factors which contribute to the potential collisions are the existence of "long moves", (in terms move

length), and the concentration of activity of the cranes in a specific area of the shop. Intuitively, when the cranes must travel a long distance while performing a move or both cranes are forced to work in a reduced area of the track, it is more probable that they will collide more frequently. The difficulty of the problems was controlled by the shape of the distribution functions used to generate the centroids and length of the moves.

For the easy problems, it was assumed that the normalized centroids (centroid divided by the length of the track) follow a Beta(1,1) distribution (same as uniform[0,1]), and the length of moves were uniformly distributed in the interval [0,18]. That is, pick-up and drop-off locations are uniformly distributed along the track. This allows the co-existence of short and long moves centered uniformly along the track which would allow the easy matching during the construction of the tours.

Difficult problems were generated using a skewed distribution of the centroids (Beta(3,1.5), skewed to the right) and length of moves in the interval [10,18]. This way at least one of the locations (lift or drop-off) is forced to be near one of the ends of the bay, and the cranes are forced to travel long distances.

In each case, centroid and length were randomly generated for each move and lift/drop-off locations were properly determined to avoid infeasibilities in the data.

Appendix A includes the listing of problem numbers with their corresponding characteristics. The specific details of problems generated for each case are discussed in the corresponding sections.

5.2 Measure of Heuristic performance

Lower Bound

Because finding the optimal solution for the problems under study is computationally infeasible, the main parameter used as comparison of performance of the heuristic was the deviation of the solution found with respect to the corresponding lower bound, called DevLB, computed as:

DevLB = 100*(Heuristic solution-LowerBound)/LowerBound

The lower bound was computed as the maximum value between:

1. The longest path in the graph P_{moves}, without considering collisions between cranes, and

2. The minimum completion time of all the moves considering the cranes as two identical machines without interference constraints, and ignoring the precedence relations among moves (if there is any).

In cases i) and ii), the lower bound was normally dictated by (2) because the longest path on the graph contains only one move while in iii) and iv) it depends on the length of the precedence chains in the graph.

Optimal Solution

Optimal solutions were attempted for the problems using complete enumeration. In essence, the procedure consists on generating all the possible assignment/sequences for the moves and cranes, and inspecting each of them with the branch and bound collision handling algorithm. This is a computation intensive process only feasible for

small size problems. Due to the computational effort required, it could only be achieved for problems of size up to 8 moves.

As a comparison, in addition to the average deviation from lower bound (DevLB) reported for the heuristic, the average deviation for the optimal solution with respect to the lower bound is also reported for problems of size 8. This information allow us to get a better idea of the quality of the results obtained and an estimate of the tightness of the lower bound used in each case.

One-Pass Heuristic

A comparison was also established with the results obtained with a simpler solution method. The benchmark used was the solution obtained with a "one-pass" heuristic where the assignment/sequencing solution was obtained the same way as described in Table 4-5, but without considering the perturbations r1 and r2. That solution was then scanned using the branch and bound procedure and the resulting solutions were used to compute the average deviations from lower bounds.

5.3 Determination of the length of the search

Preliminary runs were made to determine the impact that length of the run has on the performance of the heuristic. For case (i), easy and hard problems of size 20 were allowed to run during a long search (40,000 iterations) using different combinations of values for Weight1 and Weight2. It was found that the search progressed rapidly during the first 1,000 to 5,000 iterations and stopped making progress around iterations 10,00 to 15,000. Table 5-2 shows the results of this preliminary tests.

Perturbations Weight1/Weight2		Last Iteration Improvement found	Longest span between improvements
0.1/0.05	Maximum	15,206	11,468
	Average	8,237	4,924
	Minimum	5,126	2,843
0.1 / 0.1	Maximum	13,538	8,620
	Average	7,612	5,135
	Minimum	3,803	2,512
0.1 / 0.5	Maximum	16,628	13,597
	Average	10,182	6,916
	Minimum	3,820	2,864
0.1 / 1.0	Maximum	16,240	12,388
	Average	10,316	6,643
	Minimum	5,020	2,867

Table 5-2 Effect of length of search in performance

These results show that the major progress results in the first 10,000 iterations and continuing the search for more than, say 10,000 or 20,000 iterations does have much impact on the final results.

Having this in mind and considering that the results may be different for cases ii) through iv), the length of the search to use during the tuning and comparison phases was fixed on 20,000 iterations expecting to be checked during the experiments.

5.4 Case I: Static Problems with Independent moves

5.4.1 Generation of Test Problems

Two set of problems were generated, one for tuning purposes and the other for assessment of the quality of the solutions found. The first set contains 10 problems of size 7, 8, 9, 10, and 20 moves each, totaling 50 problems. The second set contains 10 problems each for sizes 8, 20, 30, and 40 moves totaling 40 problems. Problems of

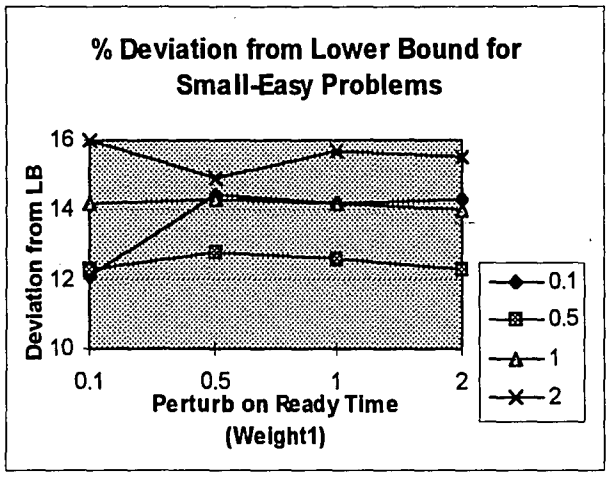
size 10 or less are distinguished as small while sizes 20 and up are recognized as big. Half of the problems in each set were considered easy and the other half were difficult.

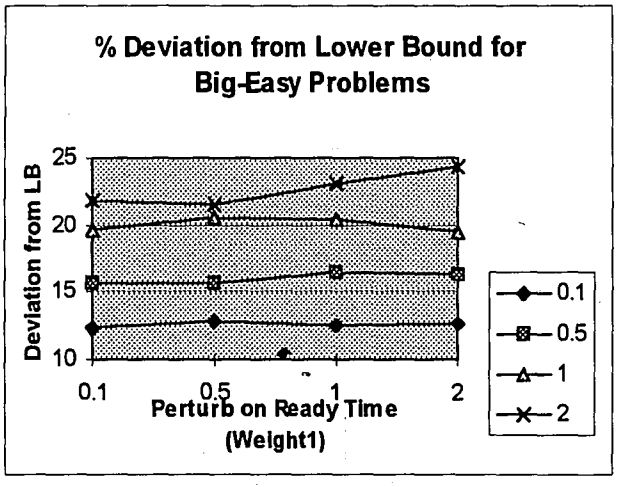
5.4.2 Tuning of parameters

Figure 5-1 shows the results of the tuning experiments as the average deviation from lower bound for problems of different difficulty in the sets 1 and 2 just described. Due to the stochastic nature of the heuristic, two replicates were run for each problem.

In this case, the quality of the solutions appears to be affected only by the perturbation on the crane assignment (value of Weight2) and not by the perturbation in the ready times for the moves (Weight1). On the other hand, the average deviations observed do not show a considerable variation for different combinations of parameters. This suggests that, for this case, the selection of Weight1 and Weight2 is not a critical decision for the performance of the heuristic.

Under these considerations, the values of Weight1 and Weight2 selected for the comparison runs were 0.1 and 0.5 , respectively, as the ones providing the best results for the hard problems of size 20.





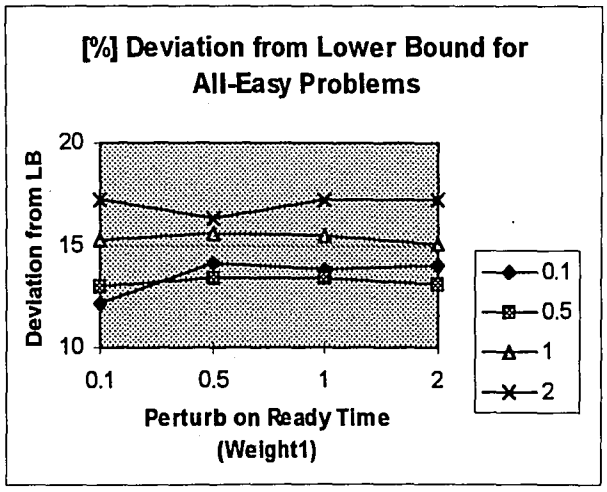
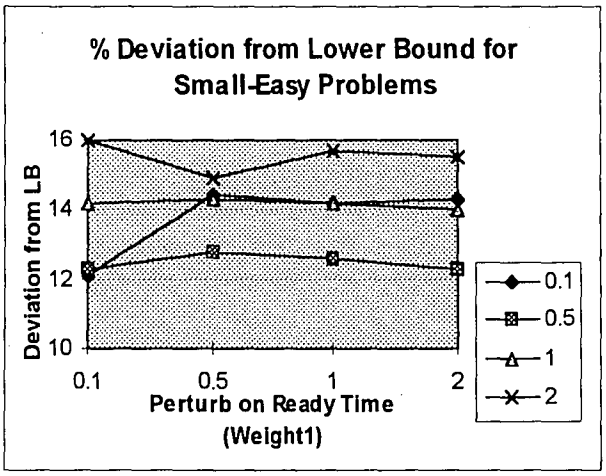
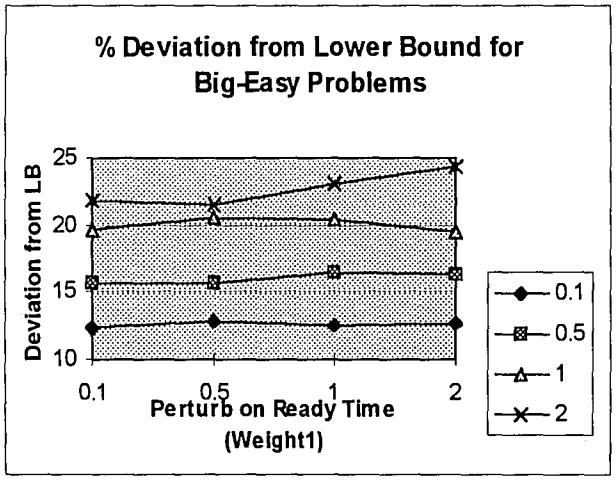


Figure 5-1 Results for tuning experiments of case I)





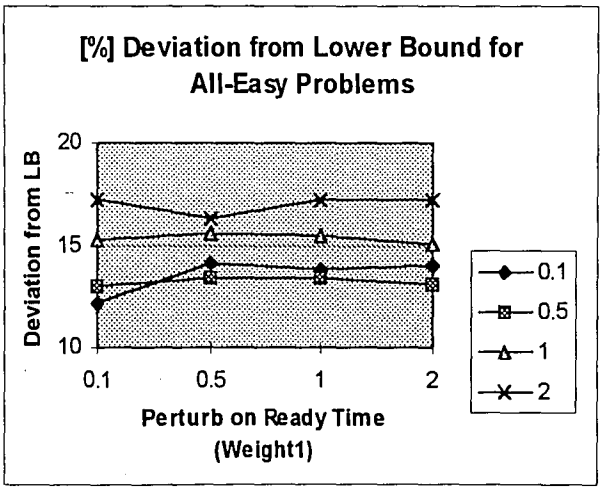


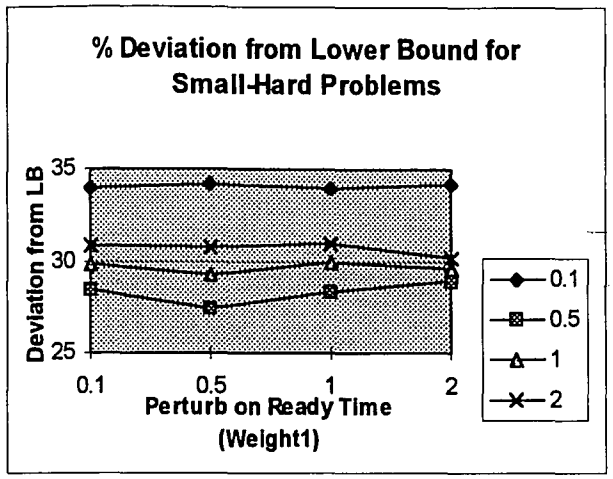
Figure 5-1 Results for tuning experiments of case I)

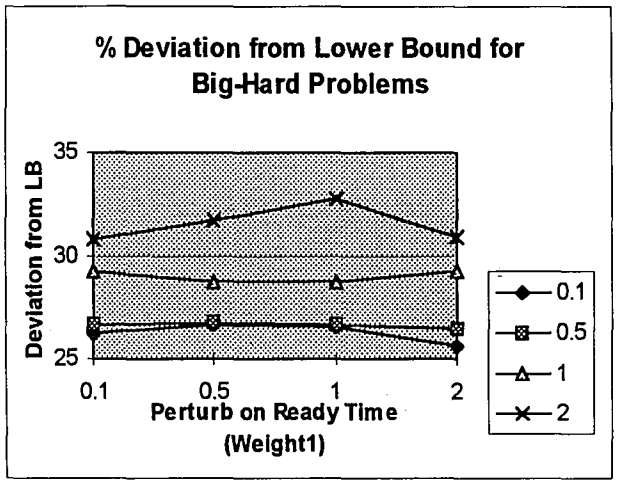
Problem Type		Small (7,8,9,10) Easy			
Deviation from Lower Bound		Perturbation on Ready Time (Weight 1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	12.1	14.4	14.2	14.3
	0.5	12.3	12.8	12.6	12.3
	1	14.2	14.3	14.2	14.0
	2	16.0	14.9	15.7	15.5

Problem Type		Big (20) Easy			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	12.3	12.8	12.5	12.7
	0.5	15.7	15.7	16.4	16.3
	1	19.6	20.6	20.4	19.4
	2	21.9	21.6	23.1	24.3

Problem Type		All (7,8,9,10,20) Easy			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	12.1	14.1	13.9	14.0
	0.5	13.0	13.4	13.4	13.1
	1	15.3	15.6	15.4	15.1
	2	17.2	16.2	17.2	17.3

Figure 5-1 (Cont.) Results for tuning experiments of case i)





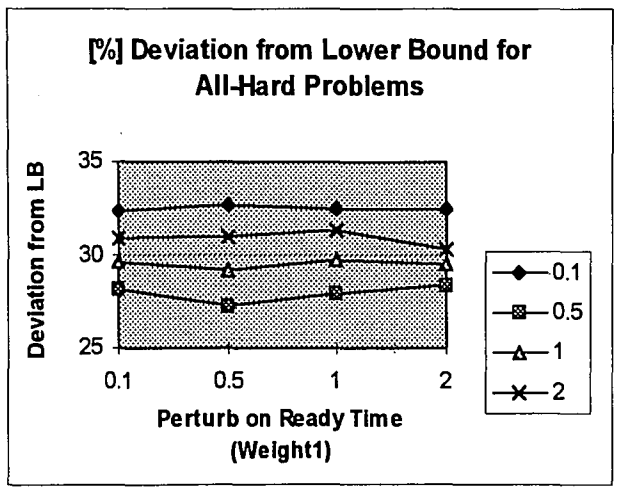
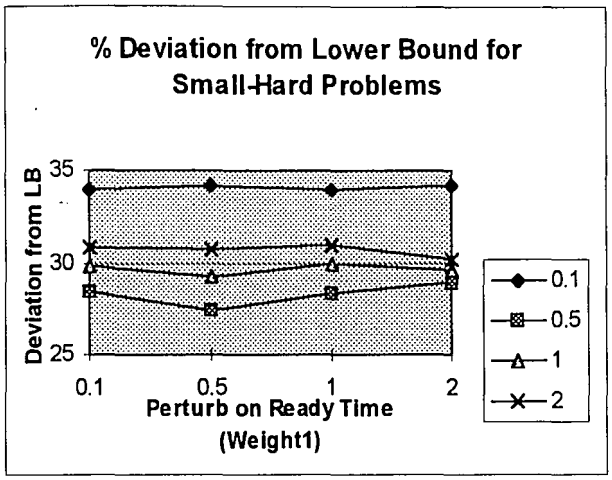
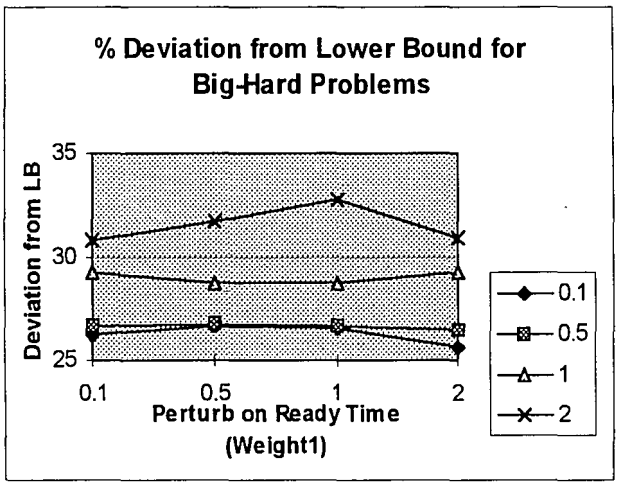


Figure 5-1 (Cont.) Results for tuning experiments of case i)





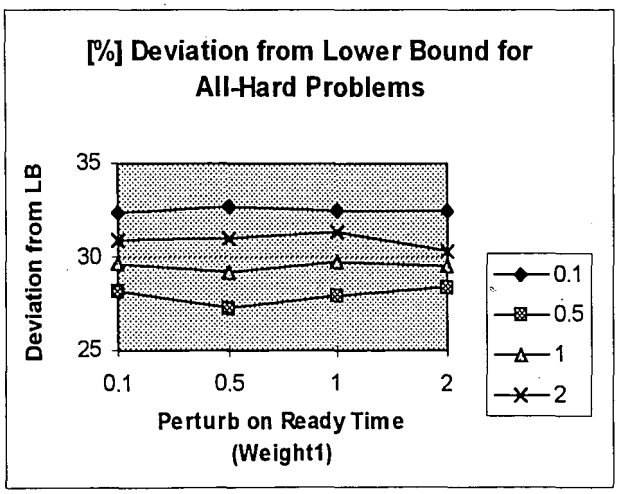


Figure 5-1 (Cont.) Results for tuning experiments of case i)

Problem Type		Small (7,8,9,10) Hard			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	34.0	34.2	34.0	34.2
	0.5	28.5	27.4	28.3	28.9
	1	29.8	29.3	30.0	29.6
	2	30.9	30.8	31.0	30.2

Problem Type		Big (20) Hard			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	26.2	26.6	26.5	25.6
	0.5	26.7	26.8	26.7	26.4
	1	29.2	28.7	28.7	29.2
	2	30.8	31.7	32.7	30.9

Problem Type		All (7,8,9,10,20) Hard			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	32.4	32.7	32.5	32.5
	0.5	28.1	27.3	28.0	28.4
	1	29.7	29.2	29.7	29.5
	2	30.9	31.0	31.3	30.3

Figure 5-1 (Cont.) Results for tuning experiments of case i)

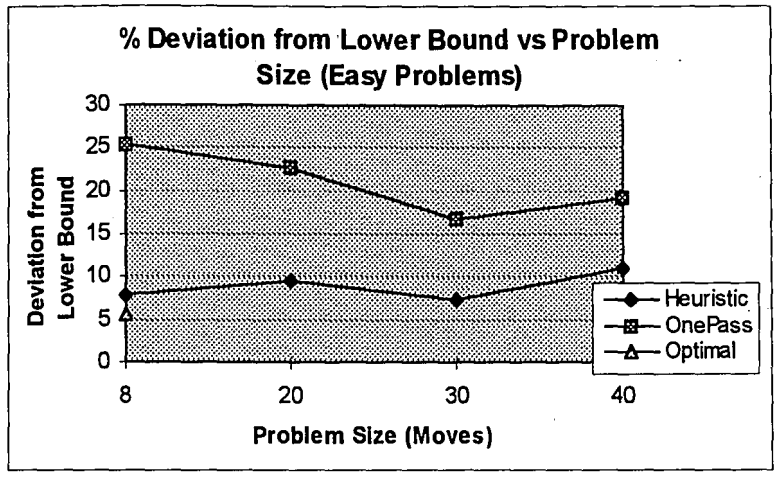
5.4.3 Comparison runs

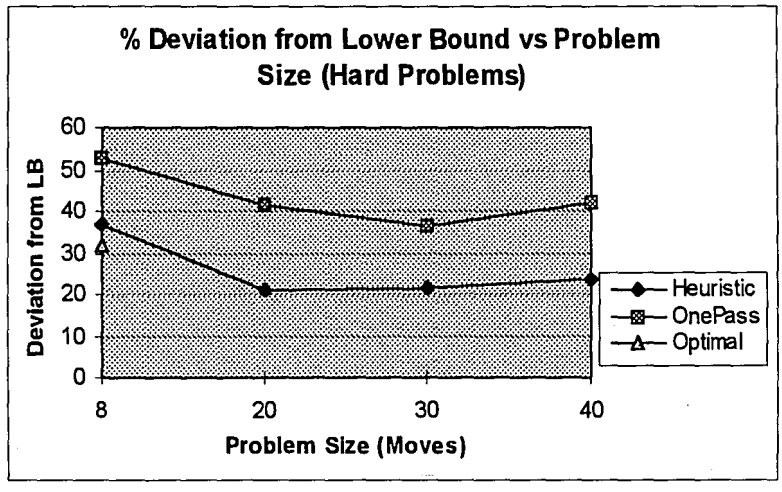
Figure 5-2 shows the result for the comparison runs considering different problem size and difficulty. Each data point represents deviation averaged from 5 problems. In other words, each plot in the figure represents 20 test problems over 4 sizes. It can be seen that the size of the problem does not have a major impact in the observed values of average DevLB. The major factor appears to be the difficulty of the problem, which produces an increase of the average DevLB from around 10% to around 20-30% when the difficulty of the problems is increased.

The results provided by the heuristic are very close to the optimal values for the small problems of size 8.

Considering the average DevLB for the optimal solutions (problems of size 8), the lower bound is loose because the optimal solutions are closer to the heuristic solutions than to the lower bound.

When compared with the one-pass method, the heuristic provides much better results. The difference becomes more evident when the optimal solutions are considered. The heuristic solutions appear now much closer to the optimal solutions than the one-pass method. The same profile is observed for the easy and hard problems.





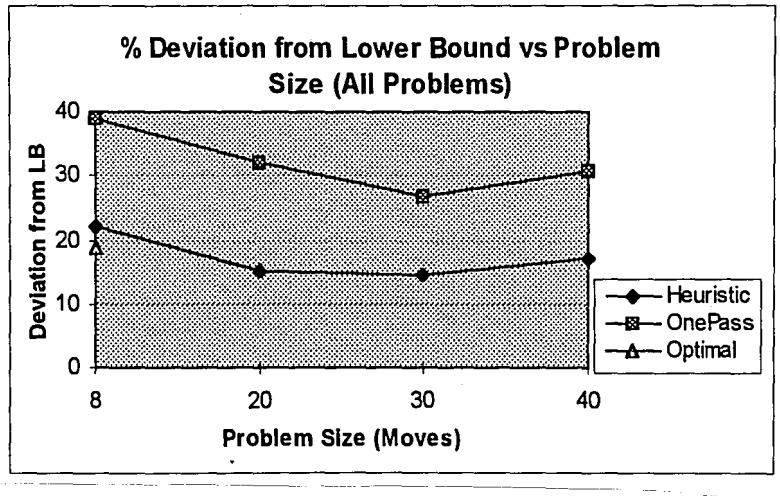
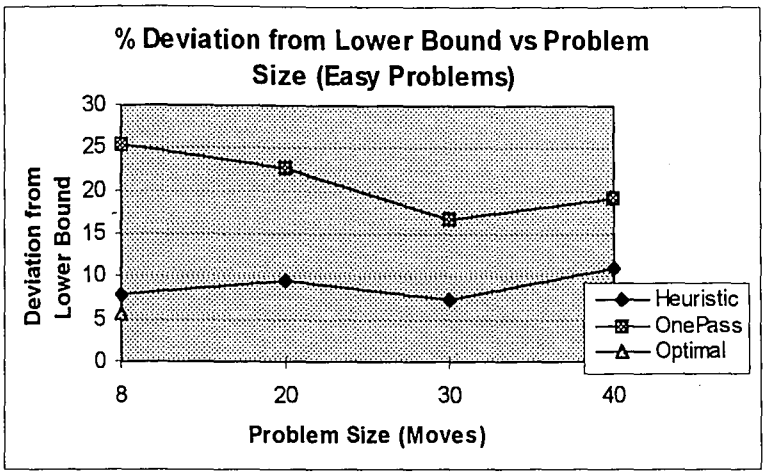
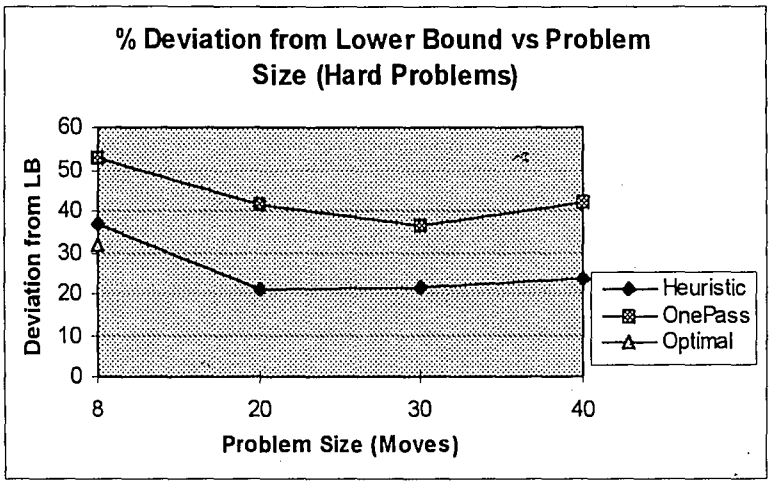


Figure 5-2 Results for comparison runs, case i).





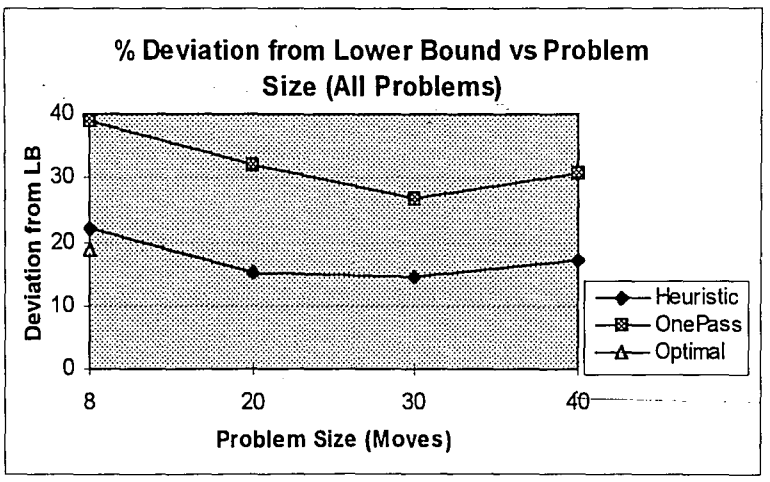


Figure 5-2 Results for comparison runs, case i).

Problem Type	Easy			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	7.89	25.39	5.6
	20	9.62	16.83	
	30	7.27	16.83	
	40	10.96	19.27	
Average		8.94	21.02	

Problem Type	Hard			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	36.67	52.72	31.94
	20	21.07	41.79	
	30	21.52	36.49	
	40	23.58	42.18	
Average		25.71	43.29	

Problem Type	Over all			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	22.28	39.06	18.77
	20	15.35	32.18	
	30	14.4	26.66	
	40	17.27	30.73	
Average		17.32	32.16	

Figure 5-2 (Cont.) Results for comparison runs, case i).

5.5 Case II: Dynamic Problems with Independent moves

As before, moves are independent, but become available at different moments in time. A ready time *t* for a move means that the lift task cannot start before time *t*, however the corresponding approach task can be initiated previously, to match the lift at time *t*.

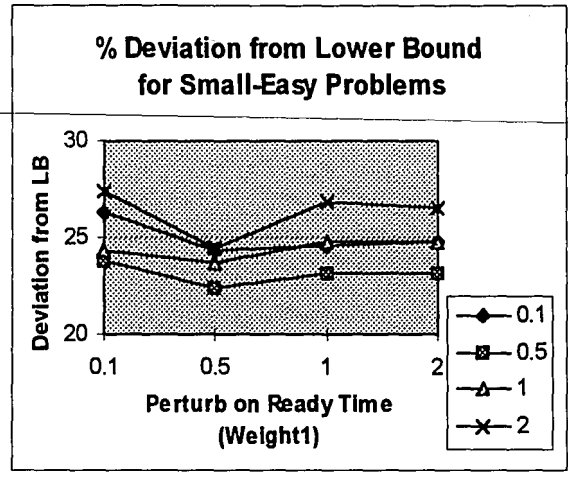
5.5.1 Generation Of Test Problems

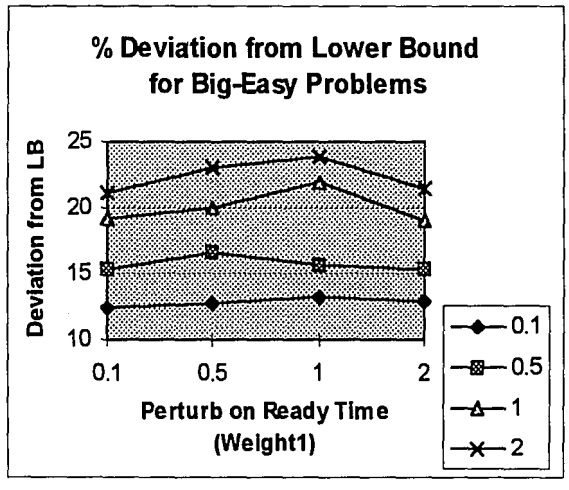
Problems were generated the same way as described in Sections 5.1 and 5.6.1 with the addition of ready times uniformly distributed in the range [0,15*Nmoves]. This range produces ready times distributed in approximately the first half or two thirds of the completion time of the moves.

5.5.2 Tuning of parameters

The same procedure as described in section 5.4.2 was used for tuning purposes. The results are shown in Figure 5-3. As before, the quality of the solutions appears to be affected only by the perturbation on the crane assignment (value of Weight2), and the average deviations observed do not show considerable variation for different combinations of parameters.

Using the same selection criteria as in case i), the values of Weight1 and Weight2 chosen for the comparison runs were 0.1 and 0.5, respectively.





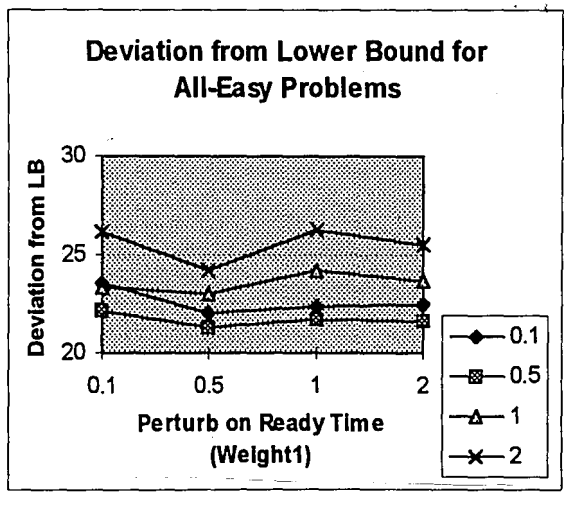


Figure 5-3 Results for tuning experiments of case ii)

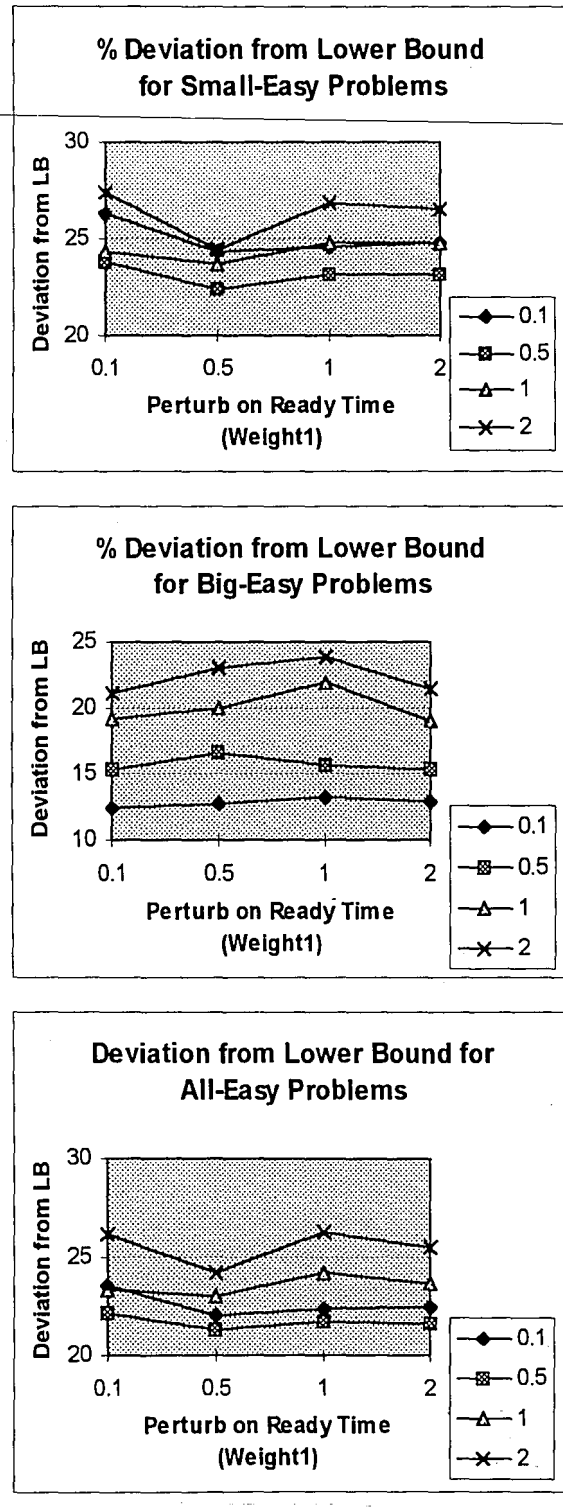


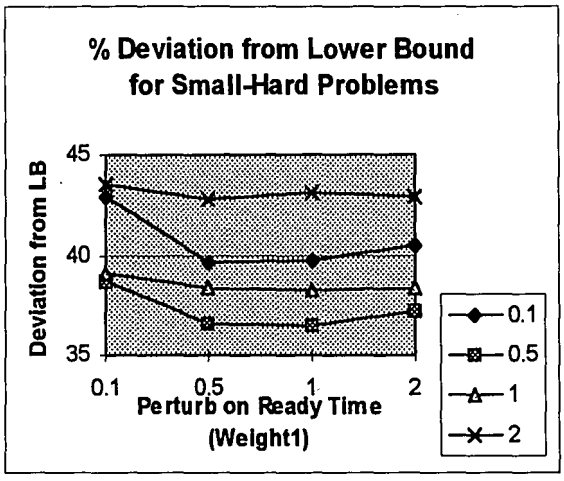
Figure 5-3 Results for tuning experiments of case ii)

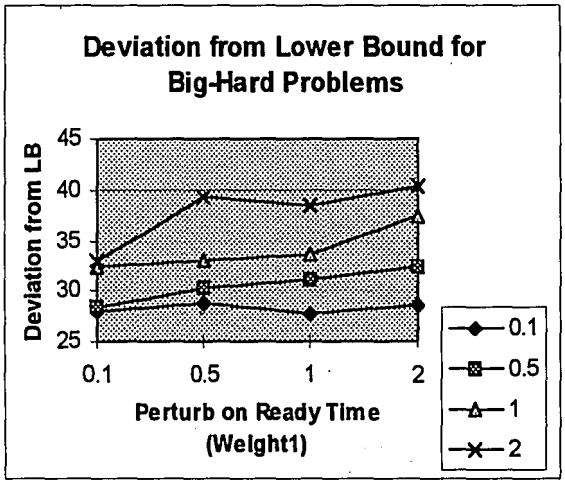
Problem Type		Small (7,8,9,10) Easy			
Deviation from Lower Bound		Perturbation on Ready Time (Weight 1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	26.3	24.4	24.6	24.8
	0.5	23.8	22.4	23.2	23.1
	1	24.3	23.7	24.8	24.8
	2	27.4	24.5	26.8	26.5

Problem Type		Big (20) Easy			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	12.4	12.7	13.2	12.9
	0.5	15.3	16.6	15.6	15.4
	1	19.2	20.0	21.9	19.1
	2	21.2	23.1	23.8	21.5

Problem Type		All (7,8,9,10,20) Easy			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	23.5	22.1	22.3	22.4
	0.5	22.1	21.2	21.7	21.6
	1	23.3	23.0	24.2	23.6
	2	26.2	24.2	26.2	25.5

Figure 5-3 (Cont.) Results for tuning experiments of case ii)





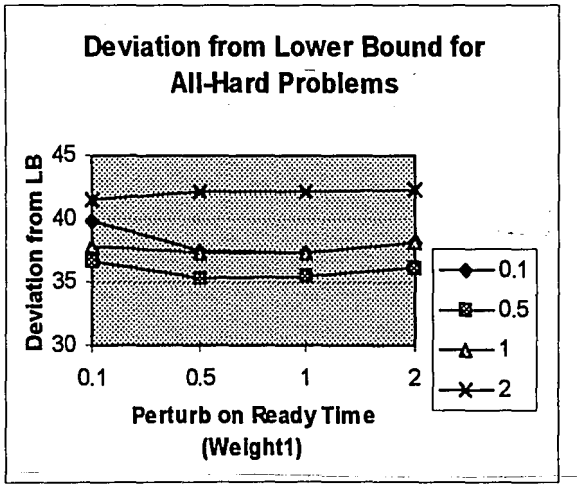
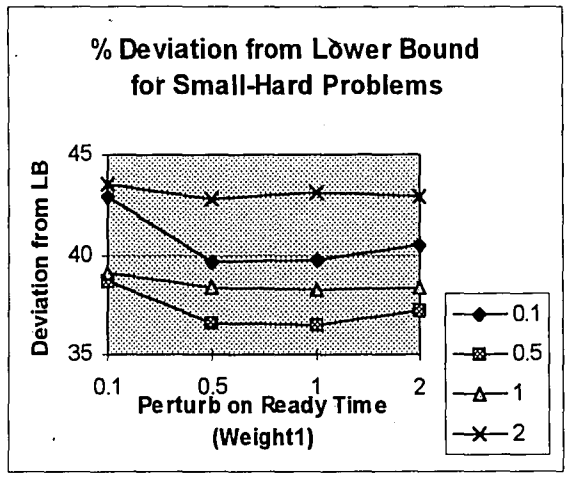
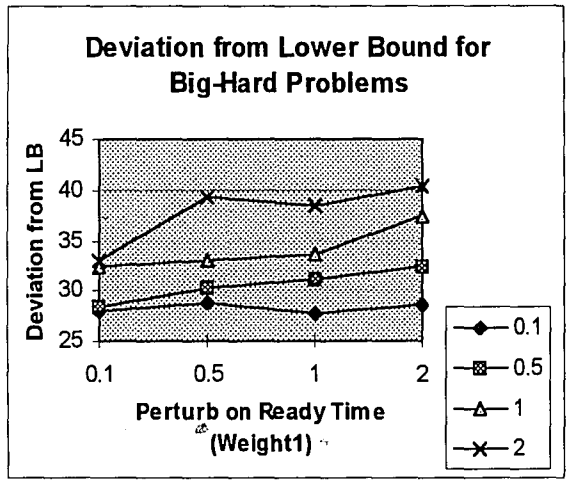


Figure 5-3 (Cont.) Results for tuning experiments of case ii)





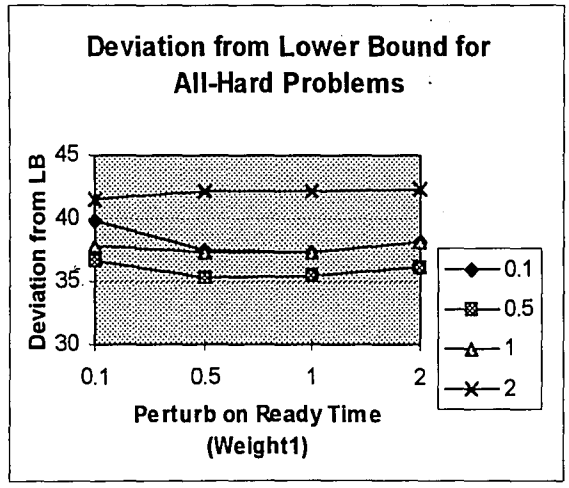


Figure 5-3 (Cont.) Results for tuning experiments of case ii)

Problem Type		Small (7,8,9,10) Hard			
Deviation from Lower Bound		Perturbation on Ready Time (Weight 1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	42.9	39.6	39.7	40.5
	0.5	38.7	36.6	36.5	37.2
	1	39.1	38.4	38.3	38.4
	2	43.5	42.8	43.1	42.9

Problem Type		Big (20) Hard			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	28.0	28.7	27.7	28.5
	0.5	28.3	30.2	31.1	32.3
	1	32.4	33.0	33.5	37.5
	2	33.1	39.4	38.5	40.3

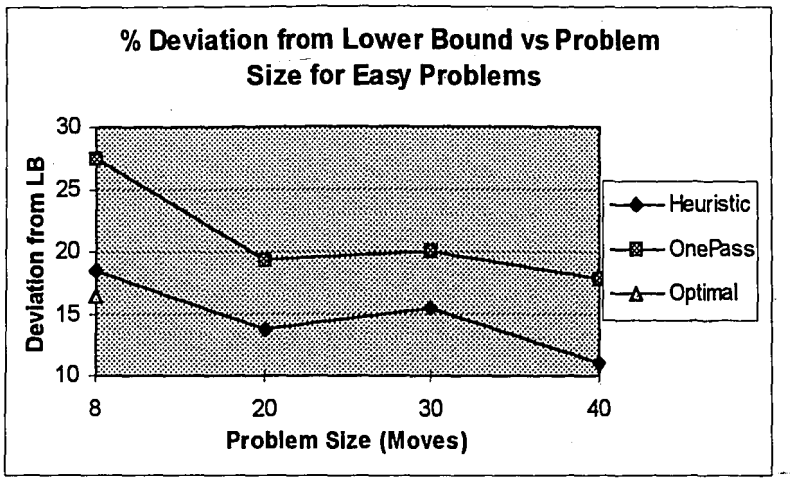
Problem Type		All (7,8,9,10,20) Hard			
Deviation from Lower Bound		Perturbation on Ready Time (Weight1)			
		0.1	0.5	1	2
Perturbation on Crane Assignment (Weight2)	0.1	39.9	37.4	37.3	38.1
	0.5	36.6	35.3	35.4	36.2
	1	37.8	37.3	37.3	38.2
	2	41.4	42.1	42.2	42.4

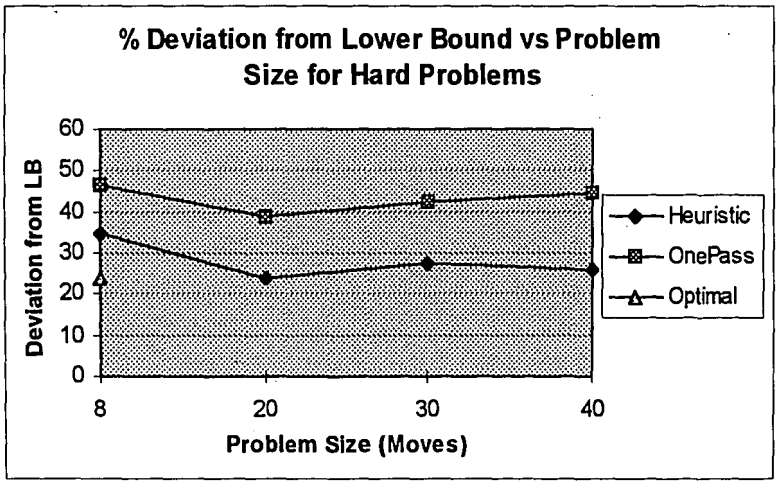
Figure 5-3 (Cont.) Results for tuning experiments of case (ii)

5.5.3 Comparison runs

As seen on Figure 5-4, the size of the problem does not have a major impact on the performance of the heuristic. Again, the difficulty of the problem has a clear effect on the lower bound producing an increase of the average DevLB from around 15% to near 30% as problem difficulty is increased. As compared with the one-pass procedure, the heuristic still produces better results. The gap between the heuristic and optimal solution is still very low (about 5% on average).

By comparing the results from the static and dynamic cases, we can conclude that there is no major change in the behavior of the heuristic both in terms of deviations from lower bound and improvements over the one-pass procedure.





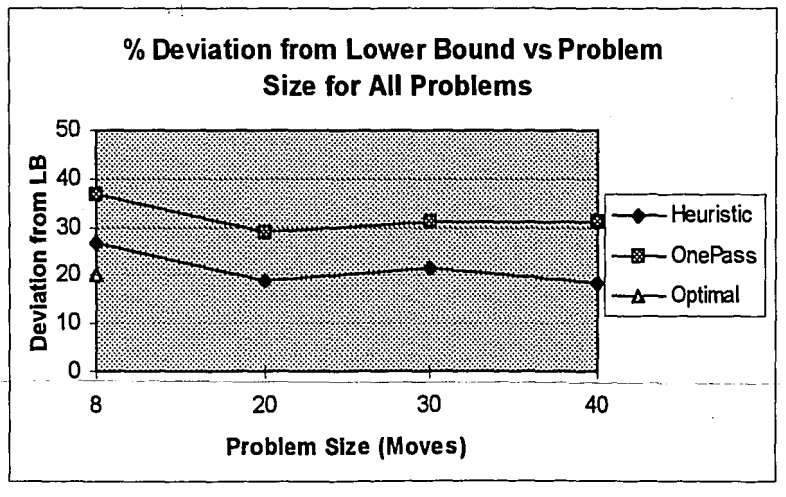
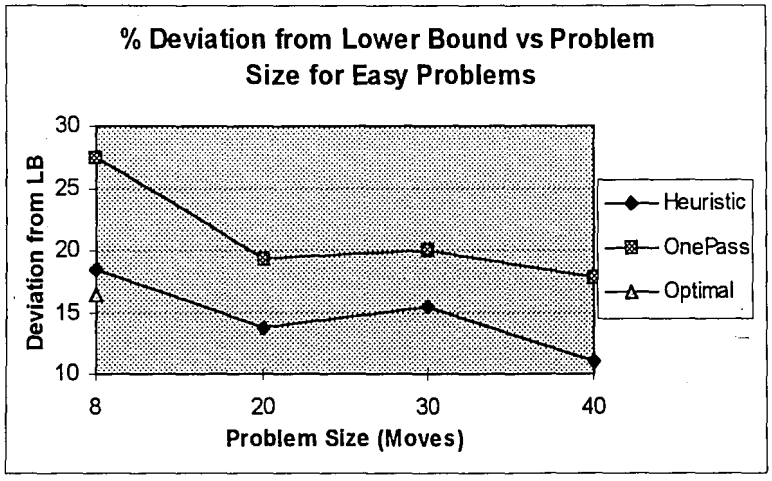
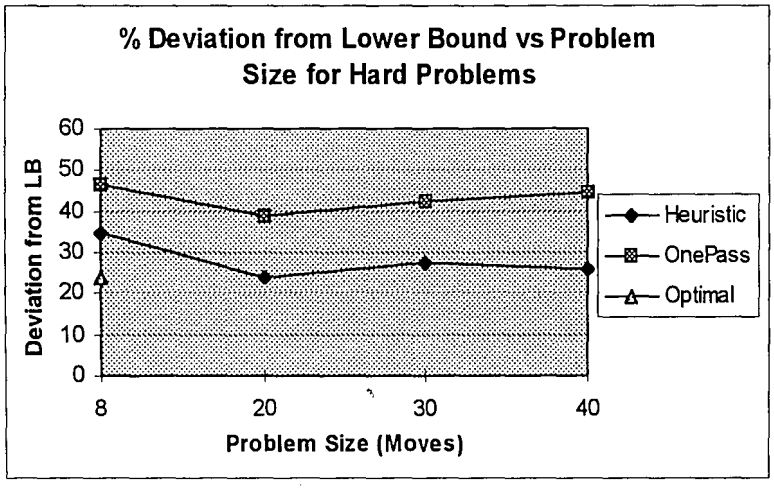


Figure 5-4 Comparison runs for case ii)





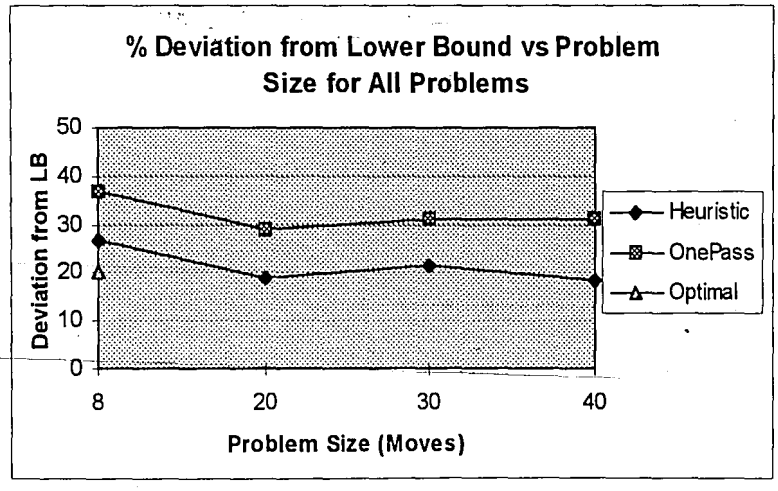


Figure 5-4 Comparison runs for case ii)

Problem Type	Easy			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	18.44	27.43	16.51
	20	13.65	19.26	
	30	15.35	20.02	
	40	10.96	17.84	
Average		14.60	21.14	

Problem Type	Hard			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	34.77	46.66	23.69
	20	23.72	38.74	
	30	27.51	42.44	
	40	25.63	44.65	
Average		27.91	43.12	

Problem Type	Over all			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	26.61	37.05	20.10
	20	18.68	29.00	
	30	21.43	31.23	
	40	18.29	31.24	
Average		21.25	32.13	

Figure 5-4 (Cont.) Comparison runs for case ii)

5.6 Case III: Job Shop with short machining times

In the general case of crane scheduling, moves corresponding to different stages of the job during its routing may appear in the same planning horizon generating precedence relations as described in Section 3.1 .

These, more general, type of problems are the focus of Sections 5.6 and 5.7. This section covers problems having machining times similar to the move times, while in Section 5.7 the machining times are much larger than the material handling times.

5.6.1 Generation Of Test Problems

The task of generating problems is now more complex than in cases (i) and (ii) because it involves: (1) creating a job shop-like problem, (2) solving a schedule, and (3) generating moves avoiding deadlocks. To facilitate the process, the same problem generator used before was transformed to recognize the generation of moves related to sequence of operations on a job. An additional simplification, made to ease the detection/resolution of deadlocks, was to consider all jobs starting and finishing at a buffer, cart, or truck area.

This simplification is not considered restrictive because of the high usage of buffer areas at BethForge, where a job rarely goes from one machine directly to the next on its routing.

Processing times for the machining operations were generated using a uniform distribution in the range [10,60] which is comparable to the move times.

Once the data for the jobshop was generated, the machine schedules were found using the Parsifal software provided by Morton and Pentico [1993], by attempting minimization of makespan as criteria. Next the solutions were manually inspected to detect machine deadlock. In such a case, additional moves are included as described in Section 3.4.

Finally, the information of precedences is included in the input files to complete the data required for each problem.

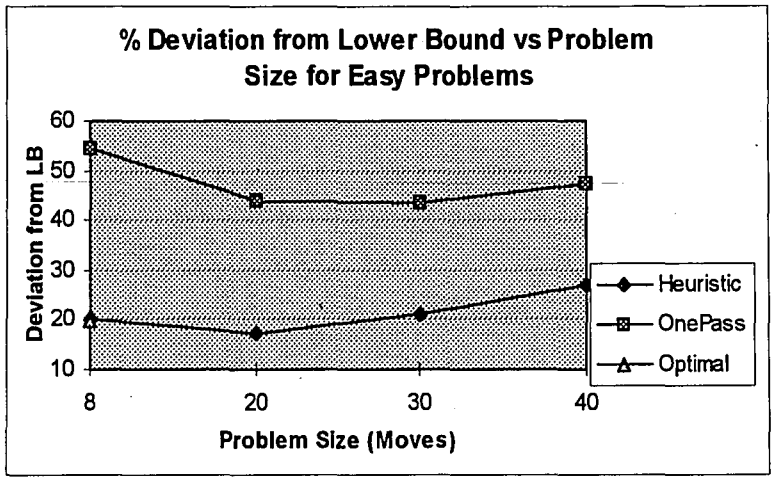
As before, easy and hard problems are generated.

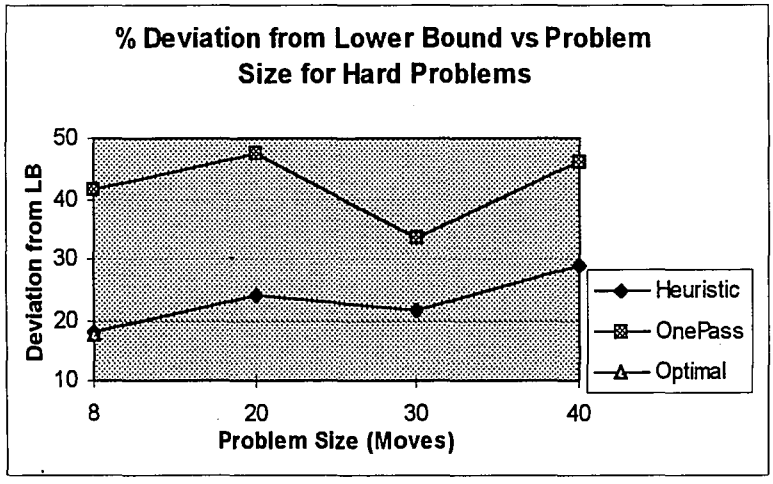
5.6.2 Tuning of parameters

As the performance of the heuristic was not very sensitive to the selection of parameters previously, it was decided to use the same values for Weight1 and Weight2 (0.1 and 0.5 respectively).

5.6.3 Comparison runs

Figure 5-5 shows the results for the comparison runs. The plots suggest that neither problem size, nor difficulty level affect the performance of the heuristic. Moreover, the results for this case look very similar to the ones obtained in case i), which may be reinforced by the fact that with the short processing times, successor moves became active very soon after being loaded into a machine providing many candidate moves for the cranes as in case I).





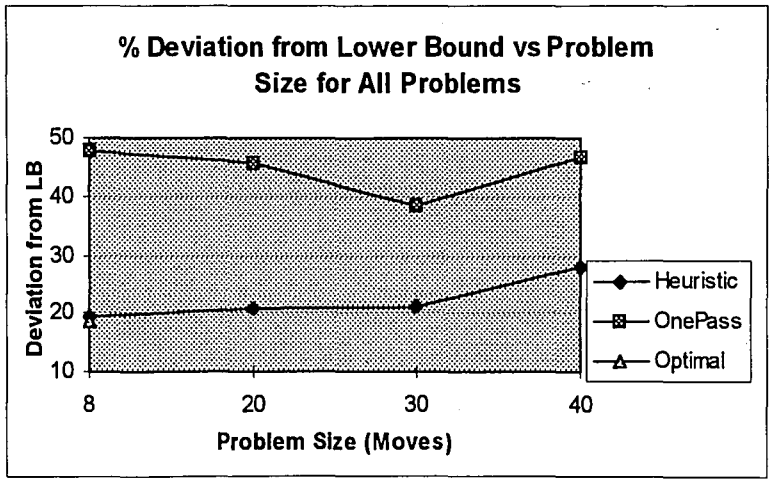
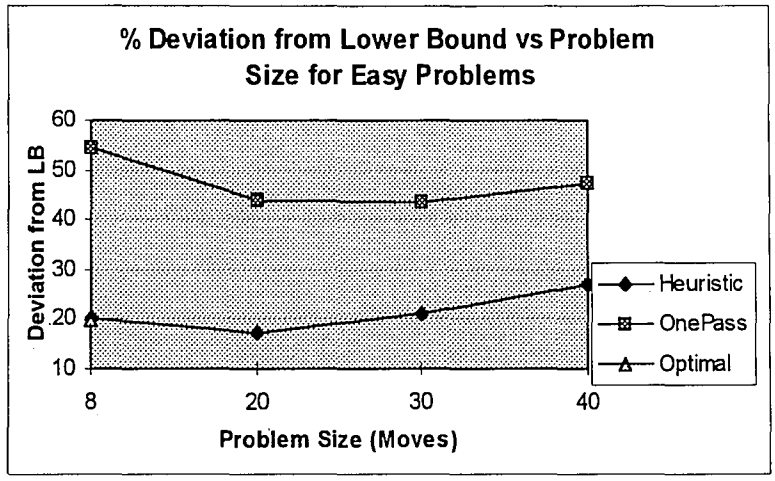
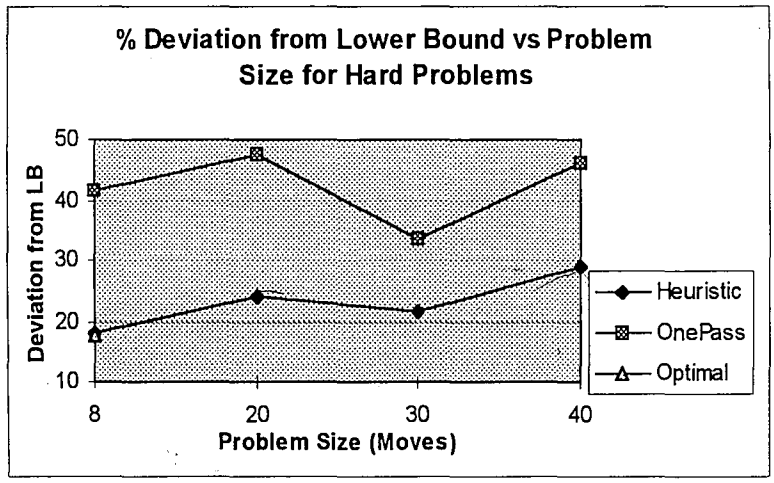


Figure 5-5 Comparison runs for case iii)





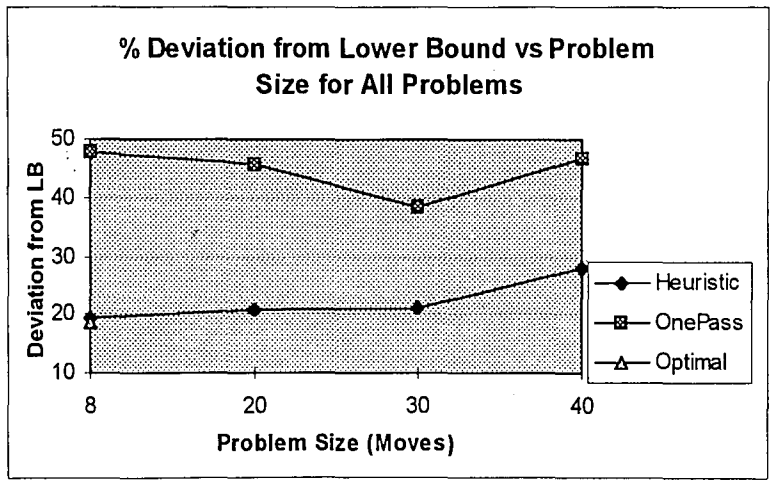


Figure 5-5 Comparison runs for case iii)

Problem Type	Easy			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	20.36	54.36	19.55
	20	17.33	43.81	
	30	20.99	43.27	
	40	26.89	47.20	
Average		21.39	47.16	

Problem Type	Hard			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	18.20	41.63	17.84
	20	24.09	47.54	
	30	21.55	33.48	
	40	29.04	45.97	
Average		23.22	42.16	

Problem Type	Over all			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	19.28	48.00	18.70
	20	20.71	45.68	
	30	21.27	38.38	
	40	27.97	46.58	
Average		22.31	44.66	

Figure 5-5 (Cont.) Comparison runs for case iii)

For the small problems, both easy and hard, the solution generated by the heuristic are, on average, less than 1% from the optimal solutions.

Again, the heuristic outperforms the one-pass procedure by a considerable margin.

5.7 Case iv: Job Shop with long machining times

5.7.1 Generation Of Test Problems

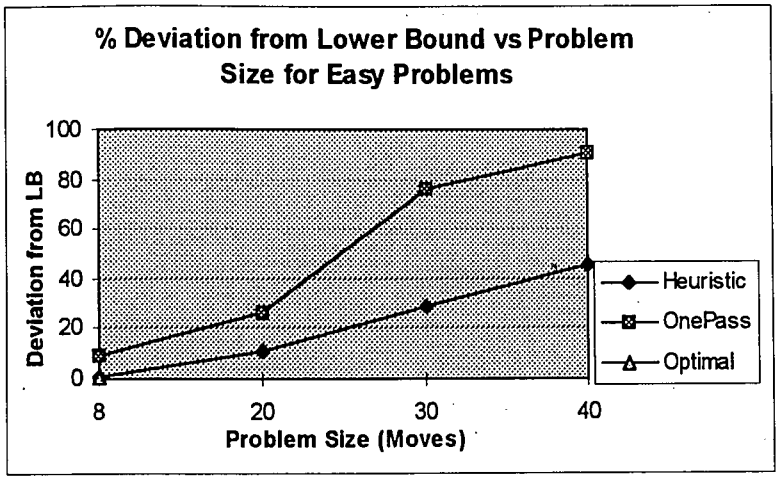
The same problems generated in the previous section were transformed by increasing the machining times by a factor of 10. All the other data for the were preserved.

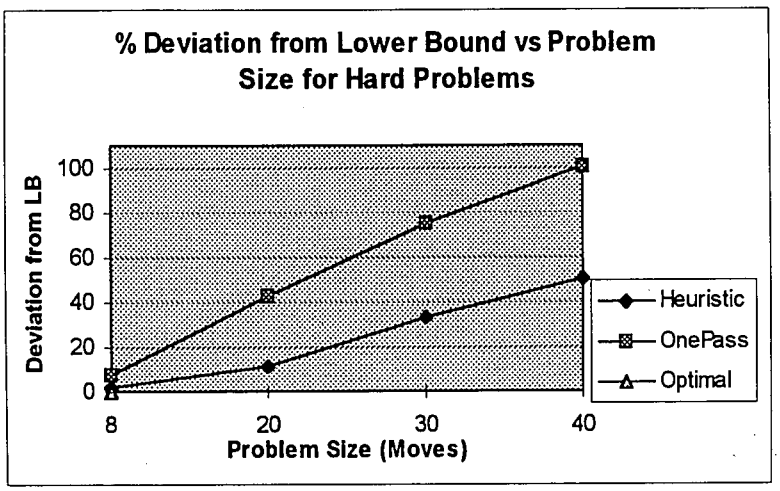
5.7.2 Tuning of parameters

The same set of parameters as in the previous cases were used.

5.7.3 Comparison runs

Figure 5-6 shows that in this case the effectiveness of the heuristic becomes more relevant as the problem size increases, both in the easy and hard problems. The improvement by using the search heuristic is remarkable for the large problems, where the one-pass procedure performs very poorly (around 100% from LB). For the small problems, the heuristic provides solutions closer than 1% from the optimal while the one-pass procedure is around 8% from optimal. In the small-easy problems, the heuristic found the optimal solution in 3 of the 5 problems.





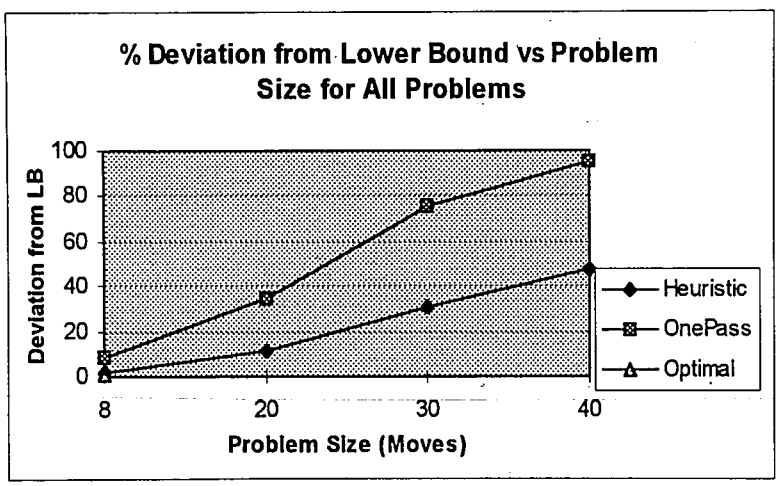
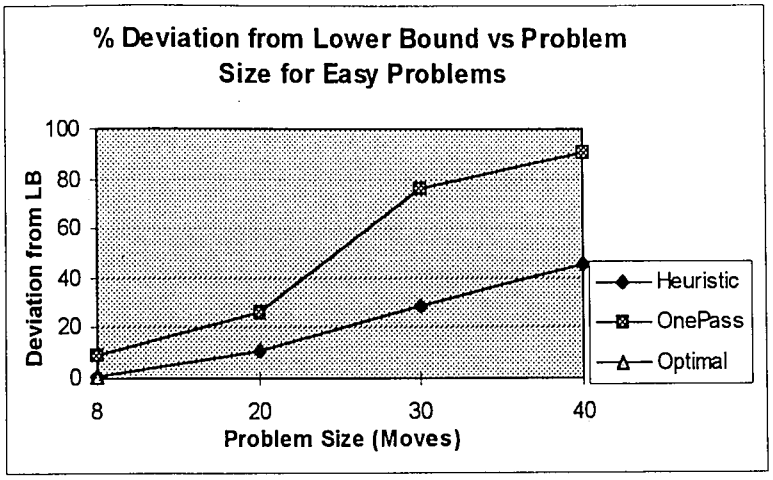
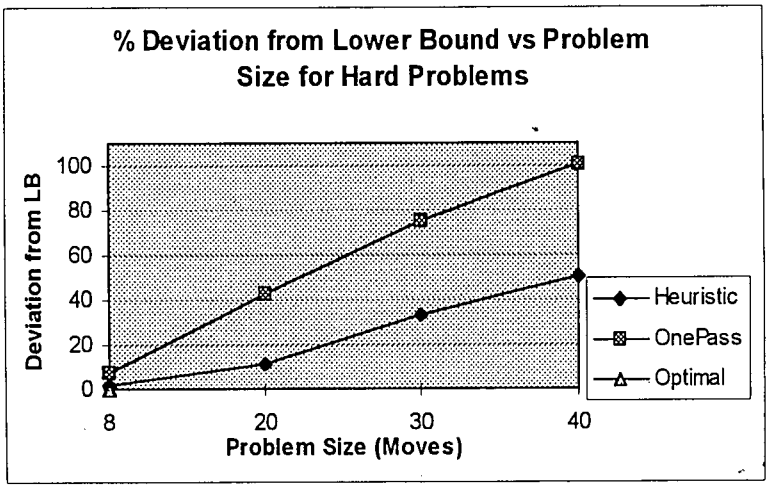


Figure 5-6 Comparison runs for case iv)





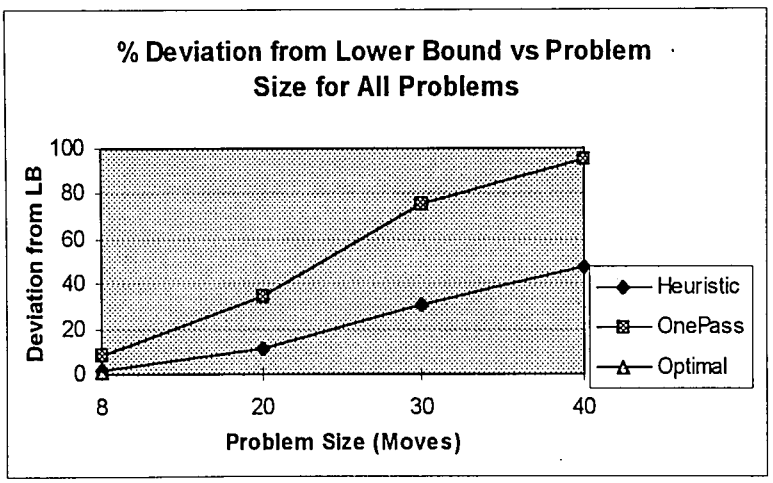


Figure 5-6 Comparison runs for case iv)

Problem Type	Easy			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	1.02	9.56	1.02
	20	10.82	26.23	
	30	28.64	76.12	
	40	45.66	90.45	
Average		21.54	50.59	

Problem Type	Hard			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	1.93	7.65	0.33
	20	11.05	42.89	
	30	32.89	74.69	
	40	50.44	100.21	
Average		24.08	56.36	

Problem Type	Over all			
Deviation from Lower Bound	Problem size	Heuristic	One-Pass	Optimal
	8	1.47	8.61	0.67
	20	10.94	34.56	
	30	30.77	75.41	
	40	48.05	95.33	
Average		22.81	53.48	

Figure 5-6 (Cont.) Comparison runs for case iv)

Chapter 6 - CONCLUSIONS

The heuristic studied in this research has proven to be powerful in terms of the quality of the solutions provided and consistency over four cases analyzed. Although some small differences in performance were observed, the selection of the parameters for the heuristic did not play a critical role during the search. Moreover, the typical profile of the search shows important improvements during the first 5,000 to 10,000 solutions inspected.

The heuristic outperforms the one-pass procedure used as benchmark, and produces solutions close to the optimal in small problems. This is a very important result pointing to the justification of using a more complex solution approach, like the one described.

The main disadvantage found during the testing with different problems has been the inherent explosion in running time as product of the branch and bound algorithm. Some additional experiments were run trying to reduce the running time by having a more strict criteria during the selection of good candidates such that during the first half of the search only "very promising" candidates were accepted. Similarly, during the enumeration procedure, new branches were created only if they may potentially produce important improvements over the current best solution. This strategy, although not fully analyzed, allowed to reduce processing times in the order of 30% without appreciable deterioration of the quality of the solutions provided.

At this point, it seems reasonable to expect that additional complexity during the generation of assignment/sequencing solutions may also help on reducing processing times by providing good candidates to the branch and bound procedure. This situation is specially important in the early stages of the search where the existence of a poor upper bound during the search mean an excessive amount of nodes of the tree being inspected.

BIBLIOGRAPHY

Armstrong, R. , S. Gu, and L. Lei, "A Greedy Algorithm to determine, the Number of Transporters in a Cyclic Electroplating Process", *IIE Transactions,* Vol. 28, No. 2, (1996), 347-355.

Armstrong, R. , L. Lei, and S. Gu, "A Bounding Scheme for deriving the Minimal Cycle Time of a Single-Transporter N-stage Process with Time Window Constraints", *European Journal of Operational Research,* Vol. 78, No. 1, (1994), 130-140.

Baker, K., *Introduction to Sequencing and Scheduling"*, John Wiley&Sons, 1974.

Conway, R., W. Maxwell, and L. Miller, *Theory Of Scheduling*, Addison-Wesley, 1967.

Daganzo, C., "The Crane Scheduling Problem", *Transportation Research-B*, Vol. 23B, No. 3 (1989), 159-175.

Eastman, W.L. , S. Even, and I.M. Isaacs, "Bounds for the Optimal Scheduling of *n* Jobs on *m* Processors," *Management Science*, Vol. 11, No. 2 (November 1964), 268-279.

Ge, Y. , "Crane Scheduling With Time Windows in Flow-Shop Environments", Ph.D. dissertation, Purdue University, 1996.

Ge,Y and Y. Yih, "Crane Scheduling with Time Windows in Circuit Board Production Lines", *International Journal of Production Research*, Vol. 33, No. 5 (1995), 1187-1199.

Graves, S. , "A Review of Production Scheduling," *Operations Research*, Vol. 29, No. 4 (July 1981), 646-675.

Han, M. and L. McGinnis, "Control of Material Handling Transporter in Automated Manufacturing", *IIE Transactions,* Vol. 21, No. 2 (June 1989), 184-190.

Lei, L. and T. Wang, "The Minimum Common-cycle Algorithm for Cyclic Scheduling of Two Material Handling Hoists with the Time Window Constraints," *Management Science*, Vol. 37, No. 12 (December 1991), 1629-1639.

Lieberman, R. and I. Turksen, "Crane Scheduling Problems," *AIIE Transactions*, Vol. 13, No. 4 (December 1981), 304-310.

Lieberman, R. and I. Turksen, "Two-Operation Crane Scheduling Problems," *AIIE Transactions*, Vol. 14, No. 3 (September 1982), 147-155.

Manier, M.A. and P. Baptiste, "A survey: The Hoist Scheduling Problem", *RAIRO Automatique-productique informatique industrielle*, Vol. 28, No. 1 (1994), 7-35.

Matsuo, H., J. Shang, and R. Sullivan,"A Knowledge-Based System for Stacker Crane Control in a Manufacturing Environment," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 5 (September/October 1989), 932-945.

McNaughton, R., "Scheduling with Deadlines and Loss Functions," *Management Science*, Vol. 6, No. 1 (October 1959), 1-12.

Morton, T. and D. Pentico, *Heuristic Scheduling Systems,* John Wiley&Sons, 1993.

Ng, W., "A Branch and Bound Algorithm for Hoist Scheduling of a Circuit Board Production Line", *International Journal of Flexible Manufacturing Systems",* Vol. 8, No. 1, (1996), 45-65.

Parker, R.G., R.H. Deane, and R.A. Holmes, "On the Use of A Vehicle Routing Algorithm for the Parallel Processor Problem with Sequence Dependent Changeover Costs," *AIIE Transactions*, Vol. 9, No. 2 (June 1977), 155-160.

Peterkofsky, R. and C. Daganzo, "A Branch and Bound Solution Method for the Crane Scheduling Problem", *Transportation Research-B,* Vol. 24B, No. 3 (1990), 159-172.

Phillips, L.W. and P.S. Hunger, "Mathematical Programming Solution of a Hoist Scheduling Program," *AIIE Transactions*, Vol. 8, No. 2 (July 1975), 219-225.

Ramaswamy, S, and S. Joshi, "Deadlock-Free Schedules for Automated Manufacturing Workstations", *IEEE Transactions On Robotics and Automation,* Vol. 12, No. 3 (June 1996), 391-400.

Shapiro, G. and H. Nuttle, "Hoist Scheduling for a PCB Electroplating Facility," *IIE Transactions*, Vol. 20, No. 2 (June 1988), 157-167.

Storer, R. , S.D. Wu, and R. Vaccari,"New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling," *Management Science*, Vol. 38, No. 10 (October 1992), 1495-1509.

Yih, Y., "An Algorithm for Hoist Scheduling Problems," *International Journal of Production Research*, Vol. 32, No. 3 (1994), 501-516.

Wysk, R.A., N.S. Yang, and S. Joshi, "Detection of Deadlocks in Flexible Manufacturing Cells", *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 6 (December 1991), 853-859.

VITA

Jaime M. Bustos was born January 27, 1965 in Temuco, Chile, to Jaime H. Bustos and Magaly M. Gomez. He graduated with highest honors in May 1986 from La Frontera University, where he earned the Bachelor of Science in Industrial Engineering. Prior to attending Lehigh University he was a faculty member in the Systems Engineering Department of La Frontera University.

END OF TITLE